Universidad de Murcia

Departamento de Ingeniería de la Información y las Comunicaciones

SISTEMAS Y ENTORNOS DE PAGO PARA LA ADQUISICIÓN DE CONTENIDOS Y SERVICIOS ELECTRÓNICOS EN RED

Tesis Doctoral

Antonio Ruiz Martínez

2009

Universidad de Murcia

D. Luis Daniel Hernández Molinero, Profesor Titular de Universidad del Área de Ciencias de la Computación e Inteligencia Artificial y director del Departamento de Ingeniería de la Información y las Comunicaciones de la Universidad de Murcia.

INFORMA:

Que la Tesis Doctoral titulada “Sistemas y Entornos de Pago para la Adquisición de Contenidos y Servicios Electrónicos en Red”, ha sido realizada por D. Antonio Ruiz Martínez, bajo la inmediata dirección y supervisión de D. Óscar Cánovas Reverte y D. Antonio Fernando Gómez Skarmeta, y que el Departamento ha dado su conformidad para que sea presentada ante la Comisión de Doctorado.

En Murcia, a 8 de Mayo de 2.009.

D. Luis Daniel Hernández Molinero

Universidad de Murcia

D. Óscar Cánovas Reverte, profesor Titular del Área de Arquitectura y Tecnología de los Computadores en el Departamento de Ingeniería y Tecnología de los Computadores de la Universidad de Murcia, y D. Antonio Fernando Gómez Skarmeta, Catedrático de Universidad del Área de Ingeniería Telemática en el Departamento de Ingeniería de la Información y las Comunicaciones de la Universidad de Murcia, AUTORIZAN:

La presentación de la Tesis Doctoral titulada “Sistemas y Entornos de Pago para la Adquisición de Contenidos y Servicios Electrónicos en Red”, realizada por D. Antonio Ruiz Martínez, bajo nuestra inmediata dirección y supervisión, y presentada para la obtención de grado de Doctor por la Universidad de Murcia.

En Murcia, 6 de Mayo de 2.009.

D. Óscar Cánovas Reverte y D. Antonio Fernando Gómez Skarmeta.

Resumen

En las comunicaciones electrónicas cada vez más sobresale la adquisición de productos y servicios electrónicos basados en pago. Sin embargo, todavía existen muchos usuarios que nos los utilizan debido a la falta de confianza que éstos generan. Uno de los principales aspectos para generar confianza en estos sistemas es la seguridad proporcionada por el sistema de pago y el uso de frameworks que, independientemente del protocolo utilizado, le permitan al usuario estar seguro de que está llevando a cabo el proceso de pago de forma correcta.

A día de hoy un amplio número de propuestas han aparecido para intentar ofrecer sistemas de pago seguros y frameworks para realizar transacciones de pago. Sin embargo, a pesar de estas propuestas, todavía quedan determinados aspectos por cubrir.

Desde el punto de vista de los sistemas de pago, las características a mejorar para los escenarios del tipo B2C (Business to Customer) de productos electrónicos son el no repudio y el intercambio equitativo. El uso de monederos electrónicos puede contribuir a incrementar la seguridad estos sistemas, facilitar la movilidad del dinero electrónico y su uso en distintos escenarios. Sin embargo, actualmente su uso para realizar pagos en la red es reducido. El principal problema que presentan las soluciones de pago basadas en monederos es la necesidad de que los distintos vendedores incorporen el uso de dispositivos hardware con módulos SAM (Secure Application Module) que pueden llevar a que el proceso de pago sea lento.

Por otro lado, cada vez existen más protocolos de pagos, con distintas características y que según éstas se adaptan mejor a la compra de unos determinados productos o servicios en determinados escenarios. Para facilitar el uso de distintos protocolos de pago en un mismo escenario aparecen los frameworks de pagos. Sin embargo, hasta ahora, no existen frameworks de pagos suficientemente genéricos que contemplen todos los elementos necesarios para utilizar estos protocolos de una forma genérica.

En respuesta a estas necesidades este trabajo de tesis presenta distintas soluciones. Por un lado presenta dos soluciones de pago basadas en monedero inteligente que pretenden facilitar el uso de estos dispositivos de una forma progresiva. En primer lugar, facilitando el pago con los monederos existentes y, en segundo lugar, proponiendo un nuevo monedero que facilite su uso para pagos en la Web.

Por otro lado se han definido distintos frameworks de pagos que permitirán la adquisición de los principales tipos de productos y servicios electrónicos: contenidos Web, contenidos distribuidos bajo streaming, servicios Web y, finalmente, servicios multimedia basados en sesiones. Estos frameworks se basan en la definición de varios componentes genéricos, como son los wallets para soportar distintos protocolos de pago, un protocolo de pagos genérico que soporta la negociación y elección de las opciones de pago a utilizar y sus precios y, finalmente, la descripción de la información de pago de una forma genérica.

De esta manera las propuestas presentadas ofrecen sistemas de pagos seguros basados en monederos electrónicos y diferentes frameworks para los principales productos y servicios electrónicos que facilitan el uso de estos protocolos así como el de cualquier otro protocolo y que pueden facilitar el uso de sistemas de pagos de un modo confiable.

Abstract

In electronic communications the purchase of products and electronic services based on payment stands out more and more. However, there are still many users that do not use them due to the lack of trust that these generate. One of the main issues to generate trust on these systems is the security provide for the payment system and the use of frameworks that, regardless of the protocol used, enables the user to be sure that he is carrying out the payment process in a correct way.

Nowadays a large number of proposals have emerged to offer secure payment systems and frameworks to perform payment transactions. However, in spite of these proposals, there are still some issues to be covered.

From the payment systems point of view, the features to improve in B2C (Business to Customer) scenarios of electronic products are non repudiation and fair exchange. The use of e-purses can contribute to increase the security of these systems, facilitating the mobility of electronic money and its use in different scenarios. However, currently its use to make payments on the Internet is reduced. The main problem that payment-based solution present is the requirement, for vendors, of incorporating hardware devices with SAM (Secure Application Module) modules that could lead the payment process is slow.

On the other hand, there are more and more payment protocols, with different features and that depending on them can be adapted better to the purchase of certain products or services in specific scenarios. With the aim of facilitating the use of different payment protocols in the same scenario, payment frameworks appeared. However, until now, there is no payment frameworks secure enough that take into account all the elements needed to use these protocols in a generic way.

As a response to these needs, this PhD thesis presents several solutions. On the one hand, we propose two payment solutions based on e-purse that intend to facilitate the use of these devices progressively. Firstly, by facilitating the payment with the existing e-purses and, secondly, by proposing a new e-purse that facilitates the use of e-purses to make payments on the Web.

On the other hand, different payment frameworks that will allow the purchase of the main types of electronic products and services (Web contents, contents distributed by streaming, Web services and multimedia services based on sessions) have been defined. These frameworks are based on the definition of several generic components, as wallets for supporting the negotiation and choice of payment options to use and their prices, and, finally, the description of payment information in a generic way.

Thus, the proposals presented offer secure payment protocols based on e-purses and different frameworks for the main electronic products and services that facilitate the use of the protocols as well as any other payment protocol. Furthermore, these proposals can improve the use of payment systems in a trusted way.

Agradecimientos

Después de muchos años, más de los que pensaba inicialmente, he llegado a escribir la última sección de esta tesis, estos agradecimientos. Durante este largo viaje, son muchas las personas que me han ayudado a prepararlo y que me han acompañado, algunos en parte del camino, otros desde el inicio al final.

Como en todo largo viaje en el que nos desplazamos a un lejano sitio y durante un largo período de tiempo, ha requerido que realice un sinfín de trabajo previo: la planificación, la reserva de los billetes, los hoteles, las maletas (que no serán ni mucho menos pequeñas), etc. En esta preparación, aparte de las personas que me han acompañado en él, hay otras que, aunque no han venido, me han apoyado, ayudado, aconsejado y puesto todo su esfuerzo para que todo me salga bien.

Cuando me acercaba a la fecha de partida, estaba ansioso e ilusionado por empezar y veía que las horas avanzan lentamente. Ansiedad que se transformó en alegría y felicidad cuando vi que el viaje empezó. Durante este viaje, conocí gente nueva que me acompañó durante parte del trayecto, algunos en cambio continuaron conmigo hasta el final.

Como en todo largo viaje hubo momentos de diversión, donde conocí a esta gente que me permitió tener otra visión del mundo y conocer otras culturas. También en las distintas etapas pude contemplar distintos paisajes, conocer distintos lugares, algunos me decepcionaron, otros me gustaron, y otros, me sorprendieron gratamente.

Pero en los viajes no todo es bonito y divertido, también surgen problemas (algunos previstos y otros que no te esperas), como la pérdida de un enlace, averías en el medio de transporte, enfermedades, etc. y hay malos momentos, momentos de aburrimiento, cansancio, enfados, disputas, compañeros de viaje que deciden no acompañarte, soledad, incomprensión, ganas de abandonar, etc. y en los que te planteas si el viaje tiene sentido y si merece la pena seguir. Sin embargo, al final, con la ayuda, el apoyo y los ánimos de las personas que te acompañan o que están al otro lado de la línea consigues resurgir cual ave fénix, recobras las ilusiones y sensaciones que te habían llevado a emprender este viaje y decides acabarlo. A partir de ahí, el tiempo pasa raudo y veloz y, finalmente, consigues llegar al final, no sin sortear algún que otro obstáculo más.

Cuando llegas de nuevo a casa, una vez superado el cansancio y, justo antes de empezar a pensar en la siguiente aventura, revives los buenos y malos momentos (aunque en mi caso, siempre hay lagunas), ves las fotos y te acuerdas de las personas que te han acompañado (aunque, desafortunadamente, no siempre de todas). Es en ese momento de recuerdo donde valoras todo lo sucedido y donde te das cuentas de lo que has sido capaz de superar con tu esfuerzo, y gracias a la ayuda de mucha gente, cada uno aportando su granito de arena, pero igualmente necesarios ya que sin la unión de todos ellos no hubiera sido posible. En mi caso también tengo que dar gracias a ese inseparable compañero que es el insomnio moderado (que aunque hace que mi procesador no sea de los más rápidos, si que le permite aumentar las horas que puede estar conectado).

Ese momento, que en parte es éste, donde rememoras y valoras todas esas vivencias y sentimientos quiero aprovecharlo para dar las gracias a todas esas personas que me han ayudado en mi largo viaje. A continuación intentaré daros las gracias a todos y cada uno de los que habéis participado en este viaje y pido perdón por aquellos de los que me pueda olvidar. Lo siento, pero ya sabéis que no es intencionado, es simplemente mi mala memoria. Así, que

¡¡¡¡¡¡¡ MUCHAS GRACIAS A TODOS !!!!!!!

De esa larga lista de personas a las que tengo que agradecer algo quiero empezar por Inma, paciente y comprensiva esposa a la que tanto quiero y de la que *I don't want miss a thing* porque me gustaría estar siempre *A tu lado*. Muchas gracias por tu cariño, compresión, ayuda, apoyo, por esas palabras de ánimo que siempre me reconfortan, por esa fe que tienes en mí, por tener que revisar gran parte de este documento (siento la tortura), aguantarme esos momentos en que estaba insoportable y un largo etcétera que valdría para escribir otra tesis. Cuando terminé el proyecto fin de carrera te prometí más tiempo y, aunque creo que lo he cumplido a medias, esta vez no cometeré el mismo error ya que, por desgracia, seguramente algunas veces caeré en las redes de una tal ANECA a la que tendré que seducir para que me deje seguir mi camino.

A mis padres, porque siempre se han sacrificado y han luchado para que estudiara, que no me faltara de nada, han intentado ayudarme en todo lo posible, me han animado cuando ha sido necesario y que, en un momento dado, cometieron la imprudencia de que un chaval tocara un ordenador (lo cuál llevó a algún que otro estropicio de vez en cuando – cuanto lo siento). En ese momento empezó mi relación con la informática, con mis primeros juegos y mis primeros programas en GW-BASIC. A partir de ahí, el camino me ha llevado hasta aquí. Espero que lo disfrutéis porque este trabajo también es parte vuestra.

A mi hermana, que siempre ha estado ahí, ha escuchado mis quejas, me ha animado y me ha ayudado en lo que ha podido para hacerme la vida más fácil, para que así pudiera dedicar todo el tiempo a estudiar, trabajar, escribir la tesis, tener un poco de tiempo libre para descansar y que fuera un poco menos vampiro. En esto de hacerme la vida un poco más fácil, Sergio también ha aportado su granito de arena.

También quiero agradecer a mis suegros, cuñados, cuñadas y resto de familia todo el ánimo y apoyo que me han dado durante este tiempo.

Otra gente que siempre me ha apoyado y animado son mis amigos de un pequeño pueblo llamado Desamparados, que durante muchos años han estado ahí, viendo como muchas veces no podía quedar con ellos y que muchas veces me han recordado que también hay que vivir un poco (nunca olvidaré carrozas, comidas, cenas y el Fenazar). Aquí también incluyo a mis compañeros de carrera y amigos de Murcia, así como los de Archena.

Agradecer a mis compañeros del equipo E-ADN-CYUM (formábamos un buen quinteto inicial): Dani Gómez, Dani Sánchez, María y Marilola, el apoyo y los ánimos que me

dieron durante los momentos difíciles cuando mi investigación en vez de ir hacia delante iba hacia atrás ya que hicieron que esos momentos fueran menos duros.

A Toñi, compañera de sufrimientos, cafés, charlas, cursos y largas esperas para ver si llega el deseado correo. Gracias por haberme liberado de muchas tareas que me han permitido centrarme y dedicar todos los esfuerzos a terminar la tesis. Nos costará, pero llegaremos, estoy convencido. A tesón seguro que no nos gana nadie.

A Sánchez, porque una parte de esta tesis se la debo a él, por lo que me ha permitido aprender junto a él y ayudarme a ver las cosas de otra forma.

Gracias también a Rafa que me ha ido aconsejando y transmitiendo su experiencia de tesis doctoral que me ha resultado de gran valor.

A Rafa, Dani Sánchez y Manolo Gil, con los que tantas cosas comparto, por estar ahí, por esos ánimos y esa ayuda que me habéis ofrecido y dado.

A PedroM y Gregorio por su apoyo y por estar siempre dispuestos a ayudarme.

To Chris Mitchell and Kostas Markantonakis from Information Security Group in Royal Holloway, University of London for their kind help in my stay there. They allowed me to learn how other people work.

A Óscar, al que tengo un buen número de cosas que agradecer ya que siempre ha estado cuando lo he necesitado y me ha apoyado en todo lo que ha podido (hay cosas que nunca se me olvidarán), del que he aprendido mucho, que ha tenido que revisar artículos desastrosos en tiempo record (sobre todo al principio), y que con su comentarios y consejos ha conseguido que mejore y que saque lo mejor de mí.

A Antonio, por confiar en mí para formar parte de este grupo, por animarme cuando las cosas no me iban bien y enseñarme a tomarme las críticas de una forma más positiva.

Finalmente, quisiera deciros, de todo corazón, que aunque vosotros podáis no verlo como yo, toda vuestra ayuda ha sido importante para mí. Por todo ello,

¡¡¡¡¡¡¡ MUCHAS GRACIAS !!!!!!!

Índice de general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

El trabajo de investigación que presentaremos a continuación está centrado en el estudio y en el desarrollo de sistemas de pago y en frameworks para tales sistemas. Nuestro objetivo es proporcionar soluciones de pago seguras que faciliten el pago de productos y/o servicios que se distribuyen de forma electrónica en entornos de comercio electrónico B2C (Business to Customer – Empresa a Cliente). Este trabajo se ha desarrollado durante los años 2001 a 2008. Durante este período de tiempo hemos podido contemplar la evolución de distintos aspectos relacionados con las tecnologías de la información y las comunicaciones y, en particular, en el comercio electrónico y los sistemas de pago para cubrir las diferentes necesidades de clientes, vendedores, bancos y proveedores de servicios de pago. Así hemos podido observar como el uso de los mecanismos de pago cada vez está más extendido y cada vez hay más soluciones. De la misma forma, también hemos podido observar cómo estas soluciones han ido evolucionado y así hemos podido obtener una visión de los diferentes problemas que han ido apareciendo y cómo éstos se pueden solucionar o se han solucionado por medio de diferentes propuestas. Sin embargo, a pesar de este amplio estudio que se ha llevado a cabo en este campo, también veremos como todavía existen diferentes problemas que deberían solucionarse para que los sistemas de pago y los frameworks respondan a las necesidades planteadas.

Durante este período de tiempo hemos propuesto distintas soluciones para cubrir las principales deficiencias que han ido apareciendo relacionadas con los mecanismos de pago para la compra de productos o servicios electrónicos. A lo largo de esta investigación, el énfasis de los aspectos a abarcar se ha modificado conforme las distintas necesidades han ido evolucionando o cambiando durante este período. De esta forma, este trabajo de investigación permite ofrecer una amplia visión de este campo y cómo es necesario ir adaptando las soluciones conforme aparecen nuevos escenarios o necesidades a cubrir.

Dentro del campo de los sistemas de pago, en este capítulo, introduciremos la problemática relacionada con los sistemas y frameworks de pago para la compra de productos y/o servicios electrónicos durante el período de tiempo mencionado. A continuación, mencionaremos los objetivos que se han ido planteando durante las distintas

fases de investigación y cómo la consecución de estos objetivos ha dado lugar a las distintas aportaciones que serán presentadas a lo largo de los distintos capítulos de esta tesis, cuya organización también será presentada aquí. Finalmente, en este capítulo presentaremos las distintas publicaciones que se han derivado de las soluciones planteadas para los objetivos establecidos.

1.1. Evolución de las TICs

Nuestra sociedad actual, conocida como la sociedad de la información, está cada vez más basada en la creación, distribución y manipulación de la información como parte de las actividades económicas y culturales. En esta sociedad, por tanto, la información es el principal elemento de producción y bien de consumo. A esta sociedad han contribuido los avances realizados en el campo de la microelectrónica y que han permitido el desarrollo de las tecnologías de la información y las comunicaciones, las conocidas como TICs (Tecnologías de la Información y las Comunicaciones). Para futuras referencias a éste o a cualquier otro acrónimo se recomienda la consulta del Apéndice A.

Estas tecnologías nos permiten cada vez procesar y almacenar más información en menor tiempo. Además, nos facilitan la posibilidad de transmitir mayor información en menor tiempo a través de los distintos avances que se producen en las redes de comunicaciones, incluso desde dispositivos inalámbricos como puede ser un ordenador portátil, una PDA (Personal Digital Assistant – Asistente Digital Personal) o un teléfono móvil. De hecho, se ha pasado a transmitir de tan sólo unos bits por segundo a transmitir millones de bits por segundo.

En esta sociedad de la información uno de sus elementos fundamentales y que se ha constituido a partir de estas tecnologías, es Internet y, en particular, la Web. De hecho cada vez es mayor el número de usuarios conectados a esta red y, también, cada vez son mayores las velocidades de conexión con las que se conectan los usuarios. Tal y como muestran los datos aportados por el Internet Systems Consortium (ISC), el número de usuarios conectados a Internet ha pasado de 72 millones en el año 2000 a alrededor de 571 millones a mediados de 2008. En España, en las mismas fechas, el número de usuarios conectados ha pasado de alrededor de 4 millones a más de 17 millones según el Estudio General de Medios (EGM). En cuanto a las velocidades, en España a día de hoy, el 87% de los usuarios conectados accede mediante tecnologías de banda ancha (según el Instituto Nacional de Estadística – INE) con una velocidad media de 3.7 Mbps según ADSLNet.

Estos datos nos muestran la importancia de esta red que inicialmente se utilizaba en el ámbito de la investigación y con propósitos no comerciales. A su uso contribuyó significativamente la aparición de la Web. Otro de los hechos que contribuyó a su extensión es que en el año 1991 se levantó la restricción existente para el uso de Internet con fines comerciales. Este hecho provocó que muchas empresas vieran una oportunidad de negocio en esta nueva red y es a partir de ahí cuando aparece el comercio electrónico en la Web.

1.2. Comercio Electrónico

A la hora de definir qué es el comercio electrónico podemos encontrar numerosas definiciones como las que aparecen en [135, 152, 290, 327]. Según Zwass [327], el comercio electrónico sería "compartir información de negocio, manteniendo las relaciones de negocio y llevando a cabo las transacciones de comercio por medio de redes de telecomunicaciones". Para Treese y Stewart [290] el comercio electrónico representa "el uso de Internet para compra y ventas de bienes y servicios, incluyendo servicios y soporte después de la venta". Finalmente, Kalakota y Whinston [152] consideran el comercio electrónico como "la entrega de información, productos/servicios, o pagos vía líneas telefónicas, redes de ordenadores u otros medios".

De las definiciones que acabamos de introducir habría que destacar varias ideas. En primer lugar, aunque una de las principales actividades en el comercio electrónico es la compra de productos en la Web, no es la única, sino que abarca más actividades como podrían ser la consulta y distribución de información de productos y catálogos, la transferencia de fondos, solicitudes de pedidos, el envío de facturas, etc. En segundo lugar, el comercio no sólo afecta a productos, sino que también intervienen servicios o información. En tercer lugar, no sólo Internet es el único medio para realizar estas transacciones sino que también podrían utilizarse otras redes de comunicaciones como podrían ser la red de telefonía básica (transacciones realizadas por teléfono fijo), redes de telefonía móvil o redes WLAN – Wireless Local Area Network – (aunque a este tipo de comercio, de forma más específica se le denominaría comercio móvil o m-commerce), bancarias o cualquier otra red de telecomunicaciones.

Entre las principales ventajas del comercio electrónico [207, 275] podemos destacar:

- Incremento de las ventas de los vendedores, ya que se pueden alcanzar segmentos de mercado más pequeños y que están dispersos geográficamente.
- Crear comunidades virtuales y conseguir nuevos clientes o proveedores.
- Descenso de costes a la hora de manejar las consultas sobre los productos y sus precios y determinar la disponibilidad de éstos.
- Se pueden disminuir los costes de transacción y, por tanto, ofrecer precios competitivos.
- Ahorro de tiempo.
- Comercios con mayor disponibilidad: 24 horas al día, 7 días a la semana.

Entre las posibles desventajas que podríamos mencionar están:

- La pérdida de la inspección física de los productos.
- La falta de entretenimiento (es menos divertido) e interacción social.
- El rápido desarrollo de las tecnologías subyacentes, que implican actualizaciones frecuentes.
- Mayor riesgo frente a fraudes y suplantación de identidad.

- La dificultad para calcular el retorno de la inversión.
- Mayor dificultad a la hora de proporcionar un servicio de atención personalizado.
- Mayor tiempo de entrega de los productos físicos que cuando esta compra se produce directamente en una tienda.

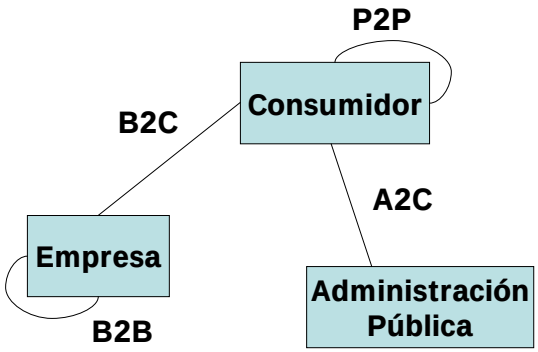


Figura 1.1. Modelos de Comercio Electrónico.

El comercio electrónico, dependiendo de la naturaleza económica de los participantes en la transacción y sus relaciones entre ellos (ver Figura 1.1), se puede clasificar en: *comercio electrónico B2C* (business to costumer), donde las actividades comerciales se llevan a cabo entre empresas y consumidores finales de una forma directa, *B2B* (business to business), donde las relaciones comerciales se llevan a cabo entre dos empresas, *A2C* (administration to consumer)/G2C (government to consumer), cuando la relación tiene lugar entre la Administración Pública y el consumidor final, y finalmente, *C2C* (customer to customer) o *P2P* (peer to peer), cuando esta relación se establece directamente entre consumidores finales.

En el comercio electrónico uno de los aspectos fundamentales es el pago por productos, servicios o información, como hemos comentado anteriormente. En general, el proceso que siguen los clientes a la hora de comprar por Internet está dividido en las fases que Maes et al. proponen en [181] (ver Figura 1.2) y que explicamos a continuación.

En primer lugar, hay una *fase de identificación de la necesidad*, donde el cliente se da cuenta de, o le surge, una necesidad. Esta necesidad también puede ser estimulada a través de información sobre productos que el cliente puede recibir por diversos medios (correo electrónico, publicidad en la Web, mailing, etc).

En la *fase de elección del producto* se realiza la evaluación de las distintas alternativas de producto. Esta evaluación está basada en los criterios proporcionados por el usuario. Como resultado de esta fase el cliente obtiene un conjunto de productos que considerará.

Una vez elegidos los productos deseados se pasa a la *fase de elección del vendedor* en la que se combinan los distintos productos elegidos junto con información específica de cada vendedor que le puede ayudar a determinar a quién comprar el producto. La evaluación de los posibles vendedores se lleva a cabo en base a una serie de criterios establecidos por el

cliente, como pueden ser precio, garantía, disponibilidad, modo de entrega, tiempo de entrega, reputación del vendedor, etc.

Una vez elegido el vendedor se puede proceder a una *fase de negociación* en la que el cliente negocia con el vendedor los términos de la transacción. Las posibilidades de negociación dependerán del tipo de mercado, producto y vendedor. Esta fase, como señalan Darko-Ampem et al. [68], Limthanmaphon et al. [172] y Röhm et al. [239], es un proceso fundamental en cualquier modelo de comercio o negocio electrónico. Por tanto, se le debe dotar de seguridad para evitar que un tercero pueda impedir que se llegue a un acuerdo satisfactorio para las partes [68].

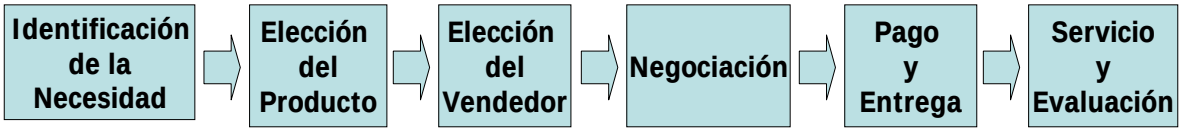


Figura 1.2. Fases del proceso de compra.

Cuando las condiciones han sido acordadas, se procede a la *fase de pago y entrega del producto*. Este proceso puede marcar la finalización de la negociación o puede ocurrir después de que la negociación haya terminado. Los mecanismos ofrecidos para realizar el pago, así como los distintos medios de entrega también pueden ser condicionantes para la anterior fase en la que se elige el vendedor con el que realizar la transacción.

Finalmente, se puede identificar una *fase de evaluación del servicio y del producto* con el fin de obtener el grado de satisfacción del cliente durante el proceso compra y de las decisiones tomadas.

De estas fases, desde el punto de vista de la seguridad del intercambio de información a través de la red, las más importantes son la de negociación, pago y entrega.

Otra forma más genérica de modelar una transacción comercial podría ser dividirla en tres fases, tal y como se propone en [239]: fase de información, fase de negociación y fase de ejecución (acuerdo o liquidación). En este caso, la fase de información se correspondería con las anteriores fases de identificación de la necesidad, elección del producto y elección del vendedor. La fase de negociación tiene la misma semántica en ambos casos. Finalmente, la fase de ejecución se corresponde con la fase de pago y entrega del producto. Sin embargo, este modelo no tiene en cuenta la posterior evaluación del grado de satisfacción de los clientes en la transacción.

Puesto que el pedido y la distribución del producto (en el caso de productos electrónicos) tienen lugar a través de una red de telecomunicaciones, el pago debería realizarse de la misma forma. Los mecanismos de pago soportados dependerán de las distintas características que se quieran ofrecer en este proceso. En la siguiente sección veremos diferentes clasificaciones dependiendo de diversos criterios.

A la hora de realizar el pago, éste se puede llevar a cabo desde un ordenador que se encuentra en una ubicación fija o se podría utilizar un dispositivo móvil, como podría ser una asistente personal o PDA (Personal Digital Assistant) o un teléfono móvil. En este último caso estaríamos hablando de lo que se conoce como *comercio electrónico móvil*,

comercio móvil o *m-commerce* [265, 272]. En caso contrario estaríamos hablando de comercio electrónico aunque, cabe destacar, que la definición genérica dada anteriormente sobre qué es el comercio electrónico abarca también el comercio electrónico móvil. Sin embargo, se establece esta distinción entre comercio electrónico y comercio electrónico móvil debido a las características peculiares de este último en cuanto al tipo y capacidades de los dispositivos, movilidad de los usuarios, etc. Las distintas diferencias existentes entre estos dos tipos de comercio aparecen descritas en más detalle en [272]. Aunque como consecuencia de los últimos avances que se están realizando en microelectrónica y que cada vez nos permiten disponer de dispositivos móviles más pequeños y con más capacidades, esta distinción podría desaparecer.

De acuerdo con estos conceptos, esta tesis se ha desarrollado dentro del comercio electrónico, y más concretamente, en el área de sistemas de pago para B2C. Nos centramos en esta área debido a varios motivos. En primer lugar, el uso de los sistemas de pago es un aspecto importante dentro del comercio electrónico, especialmente, en el comercio B2C. En segundo lugar, cada vez se realizan más transacciones de este tipo en la red y los distintos vendedores quieren ofrecer sus servicios/productos a través de la red y ser pagados por ellos de forma electrónica. Por último, a pesar de que existen soluciones para realizar este tipo de transacciones, todavía podemos encontrar distintos aspectos por solucionar y/o mejorar en cuanto a la seguridad y movilidad de la transacción así como la distribución del producto. Un análisis más detallado se presenta en el Capítulo 2.

El comercio electrónico alcanzó su máximo auge, en cuanto a la creación de empresas con fines comerciales en Internet y la realización de propuestas relacionados con este campo, a mediados de los 90, más concretamente, entre los años 1995 y 1998. Esta época constituyó la edad de oro del comercio electrónico tal y como Tian y Stewart [287] la han denominado. Durante este periodo aparecieron algunas de las propuestas más importantes en cuanto a protocolos de pago (Millicent, Payword, SET, etc) así como algunas de las empresas de comercio electrónico más importantes como Amazon o eBay.

Posteriormente, entre los años 2000 y 2001 se produjo la explosión de la burbuja de las punto-com, en la que muchas de las empresas fueron a la bancarrota. La explicación se debe a que se crearon unas expectativas que no eran realistas en cuanto a las compañías de comercio electrónico e Internet.

A partir del año 2002, esta crisis permitió llevar a cabo un estudio más realista de la situación actual y de las necesidades a cubrir. A partir de ahí, el comercio electrónico empezó a resurgir y actualmente, el comercio electrónico empieza a presentar un futuro importante [173] una vez que las tecnologías de Internet están más asentadas y existe una base importante de usuarios en Internet. De hecho, en Estados Unidos las ventas mediante comercio electrónico han aumentado en un 13.6% (±1.2%) [299] desde el primer cuatrimestre del año 2007 al primero del 2008. En España, ese crecimiento se eleva a un 39% de 2007 a 2008 [59]. Sin embargo, para que el comercio electrónico siga creciendo es necesario el desarrollo de algunos aspectos que lo limitan como son el acceso universal, aspectos de seguridad y privacidad, y el fraude en Internet [287]. Más en concreto, como se mencionan Gefen et al. en [118] y Shang et al. en [275], los principales problemas que presenta a día de hoy el comercio electrónico son: la falta de confianza del usuario en los

sistemas de pago, sistemas complejos de manejar, interfaces no comunes y/o amigables, seguridad de los sistemas, falta de pilotos que muestren al usuario el funcionamiento de los sistemas de comercio, metodologías para analizar y modelar la seguridad de las transacciones de negocio.

Los datos presentados muestran que el comercio electrónico, y en particular, el que se produce entre las empresas y los usuarios finales (denominado comercio electrónico B2C) es uno de los principales ámbitos que se han beneficiado de Internet debido a las ventajas que éste ofrece y que fueron comentadas al principio de esta sección.

Dentro del comercio electrónico, uno de los principales aspectos es el pago por productos o servicios. Estos pagos de productos o servicios se llevan a cabo por medio de los sistemas de pago electrónico y protocolos de pago.

1.3. Sistemas de pago electrónico

El objetivo de los sistemas de pago es permitir la realización de pagos a través de una red de telecomunicaciones. Estos pagos electrónicos se pueden realizar por productos físicos, por servicios o por productos/servicios electrónicos. Este último tipo de productos/servicios está cobrando una especial importancia en los últimos años debido a la aparición de una amplia gama de productos digitales tales como revistas electrónicas, archivos musicales, vídeos, etc., así como la aparición de servicios electrónicos tales como VoIP, juegos en red, etc. En el Capítulo 2 se presenta una caracterización de los sistemas de pago más detallada.

En respuesta a la necesidad de realizar de pagos en la red han aparecido multitud de propuestas de pago que intentan llevar a cabo este proceso de una forma segura en distintos tipos de escenarios. De hecho, en el año 2002 ya se contabilizaban más de 200 propuestas de pago [40]. Posteriormente, en el Capítulo 2 analizaremos las propuestas más relevantes.

La seguridad es una de las principales características que tiene que ofrecer un sistema para generar confianza en los usuarios. Sin embargo, muchas de estas propuestas ofrecen una seguridad limitada y no tienen en cuenta aspectos tales como el no repudio, el intercambio equitativo, la atomicidad, no están validadas formalmente y utilizan como método de pago las tarjetas de banda magnética (crédito o débito). El principal problema de las tarjetas de banda magnética es que el nivel de seguridad que ofrecen es reducido. Cualquiera que posea el equipo necesario puede leer, borrar o reescribir la información almacenada en la banda de forma que es posible clonar este tipo de tarjeta. Otro de los problemas que presentan es que, a partir de la información de nuestra tarjeta, es posible realizar, sin ningún tipo de problema, transacciones en la red sin tener que llevar a cabo ningún proceso de autenticación.

En respuesta a los problemas que presentan las tarjetas de crédito basadas en banda magnética aparecen las tarjetas inteligentes y los monederos electrónicos para proveer de un mayor nivel de seguridad a la información de pago.

Aparte de la seguridad, una de las principales ventajas de los monederos electrónicos es que facilitan la movilidad ya que el dinero electrónico contenido o gestionado por ellas se

transporta en un dispositivo que es como una tarjeta de crédito. Los monederos electrónicos se pueden utilizar para realizar tanto pagos en local como en remoto aunque, a día de hoy, principalmente se utilizan para realizar pagos en local (en máquinas de vending). En el Capítulo 2 analizaremos más en detalle las tarjetas inteligentes y los monederos electrónicos.

A pesar de estas ventajas que acabamos de mencionar, los monederos electrónicos no están muy extendidos y su uso en transacciones de Internet es casi inexistente. Las causas que explican estos hechos son varias. En primer lugar cabe mencionar que hasta ahora el uso de las tarjetas inteligentes era poco conocido y estaba poco extendido ya que se trataba de un dispositivo más caro que una banda magnética. A día de hoy este problema empieza a ser menos significativo ya que está produciendo la migración de las tarjetas de banda magnética a este tipo de tarjeta. Además, gracias a la regulación establecida por determinados países (principalmente Europa y Estados Unidos), las tarjetas inteligentes con determinadas características pueden ser consideradas un dispositivo seguro de creación de firma. De esta forma la firma electrónica puede ser considerada equivalente a la firma manuscrita. Por esta razón los distintos gobiernos han empezado a utilizarlas como dispositivo de identificación nacional electrónico. Otro de los problemas que ha derivado en que no se utilicen los monederos como el estándar CEN EN 1546 (Monedero electrónico inter-sector o más conocido simplemente como WG10 [44]), CEPS [45] (Common Electronic Purse Specifications – Especificaciones de monedero electrónico común) y EMV (Europay, MasterCard, Visa) [89, 90, 91, 92] para realizar pagos en la Web es la necesidad de que los vendedores integren en su aplicaciones dispositivos con módulos SAM (Secure Application Module – Módulo de Aplicación Segura) que permitan aceptar pagos basados en monedero. Aparte de esta integración, esta comunicación entre dispositivos requiere el intercambio de varios mensajes lo que puede llevar a que la comunicación sea lenta. A día de hoy el uso de los monederos en pagos en Internet presenta un interesante reto desde el punto de vista de la investigación.

Aparte de las características de seguridad, otras características interesantes que deberían soportar los protocolos de pagos sería tener en cuenta la seguridad en las principales fases en el proceso de compra de un producto como son la negociación del precio y las características, el pago y la distribución de los contenidos de forma segura.

Con el fin de abordar esta problemática y ofrecer protocolos más seguros y que cubran distintos ámbitos o modelos de negocio, conforme pasa el tiempo, aparecen más sistemas de pago. En un momento dado, a la hora de realizar un pago, un cliente y un vendedor podrían soportar distintos mecanismos de pago en común. En este escenario se debería facilitar a clientes y vendedores la elección y el uso de un protocolo de entre las opciones disponibles. Los frameworks de pago que se introducirán en la siguiente sección nos van a permitir llevar a cabo este proceso.

1.4. Frameworks de pago electrónico

Los frameworks de pago nacen de la necesidad de ofrecer una solución genérica de pagos que permita el intercambio y la selección del protocolo de pago a utilizar en una determinada transacción. Estos frameworks pretenden que, a pesar de las diferencias que

presentan los distintos protocolos, su uso para la realización de pagos sea sencillo, transparente y uniforme. Tal y como mencionan Gefen et al. en [112] y Wareham et al. en [287] se debe ofrecer un mecanismo uniforme que genere confianza en los usuarios. Es importante que para los usuarios, independientemente del protocolo utilizado, el proceso de pago siga los mismos pasos de forma que el usuario esté seguro de que está realizando el proceso de forma correcta.

Por tanto, el objetivo de los frameworks de pago es facilitarnos la realización de pagos en Internet para los distintos tipos de productos y/o servicios que queramos adquirir. Para un framework de este tipo será necesario definir varios componentes: un mecanismo para describir la información de pago e incorporarla en la Web, un método para intercambiar la información de los métodos de pago soportados y poder elegir uno y, finalmente, un mecanismo para soportar cualquier protocolo de pago y poder intercambiar los mensajes. Sin embargo, a día de hoy las soluciones propuestas (IOTP [32, 33], SEMPER [165, 166], etc.) no cubren todas estas necesidades y sólo se centran en alguno de estos aspectos. Estos frameworks de pagos y sus principales características serán analizados más en detalle en el Capítulo 2.

1.5. Objetivos

Una vez vista la problemática existente relacionada con los sistemas de pago electrónico y con los frameworks de pago pasaremos a describir los objetivos que nos planteamos resolver en esta tesis. El objetivo genérico sería ofrecer soluciones de pago seguras y frameworks de pago para la adquisición de productos y/o servicios electrónicos en entornos de comercio electrónico B2C. Este objetivo genérico a su vez se puede dividir en una serie de subobjetivos. Así, los principales objetivos que se plantean para este trabajo de investigación son los siguientes:

- Definir soluciones de pago para la adquisición de productos electrónicos en entornos de comercio electrónico B2C que garanticen la seguridad de la transacción, el intercambio equitativo y la atomicidad.
- Las soluciones de pago estarán basadas en el uso del monedero electrónico ya que facilita la movilidad del dinero electrónico del usuario y proporciona mayor seguridad que las tarjetas de crédito. Además, facilitarán su uso en distintos escenarios como pagos en máquinas de vending (pagos locales), pagos en Internet, etc.
- Se buscará facilitar a los vendedores el soporte de soluciones de pago basadas en monedero. Por tanto, se tratará de evitar que los vendedores tengan que integrar dispositivos como módulos SAM. Esta integración puede ser compleja en las aplicaciones, puede requerir un elevado número de estos dispositivos si se quieren soportar muchas transacciones simultáneas y puede producir que la realización del pago sea más lenta debido a los mensajes que hay que intercambiar entre el SAM y el monedero.

- Se considera la seguridad en las principales fases que forman parte de una transacción de compra: la negociación del precio y las características, el pago y, finalmente, la distribución del producto.
- Una vez abordado el problema de los sistemas de pago seguros, buscaremos ofrecer un framework (o varios) de pago que permita la utilización, la negociación y el uso de distintos protocolos de pago a la hora de realizar compras en la Web para los principales tipos de productos y/o servicios electrónicos: contenidos de páginas HTML, contenidos que se distribuyen bajo streaming, servicios/contenidos que se distribuyen mediante el establecimiento de sesiones multimedia y, finalmente, servicios Web.
- Los componentes definidos para el framework debería ser genéricos de forma que puedan soportar cualquier protocolo de pagos de una manera uniforme.
- Con el fin de facilitar la búsqueda y descubrimiento de la información de pago, buscaremos definir la información relacionada con el pago de forma que sea procesable automáticamente.

1.6. Contribuciones de la tesis

En respuesta a los objetivos definidos en la anterior sección, en esta tesis se presentarán diferentes propuestas. En la Figura 1.3 se muestran conceptualmente las soluciones aportadas y las relaciones existentes entre ellas. Como también se puede ver en la figura, las soluciones se encuentran a diferentes niveles, tratando de delimitar los distintos niveles relacionados con la problemática de los sistemas de pago.

En el nivel más bajo, como base para el resto de aportaciones, podemos encontrar las soluciones de pago. En esta tesis vamos a presentar dos soluciones basadas en monedero: SPEED [249] y PURSE-COIN [251, 252].

SPEED [249] es un protocolo de pago basado en monedero electrónico que garantiza la negociación segura y el intercambio equitativo. El producto se entrega de forma cifrada al cliente de forma que ningún intruso en la comunicación pueda obtenerlo. Esta solución está pensada para facilitar a los vendedores la aceptación de pagos basada en los monederos existentes como WG10, CEPS y EMV que requieren que el receptor del pago incluya en su sistema un dispositivo con un módulo SAM. Para evitar esta integración al vendedor, esta tarea se delega en un broker que además actúa como tercero de confianza garantizando el intercambio equitativo.

Si bien es cierto que con SPEED facilitamos la aceptación de pagos basados en monedero al vendedor, también es cierto que la comunicación entre el monedero y el SAM sigue siendo necesaria. Esta comunicación puede llevar a que la realización de pagos sea lenta, además de que requiere la participación de un tercero en todas las transacciones. Con PURSE-COIN [251, 252] buscamos ir un paso más allá y eliminar la necesidad de esta conexión, facilitando aún más si cabe el uso de monederos. De hecho esta propuesta trata de combinar las ventajas de los monederos con las ventajas de las monedas electrónicas. En consecuencia, en PURSE-COIN a la hora de realizar el pago, el monedero genera una moneda específica para un vendedor. Esta moneda puede ser intercambiada

con el vendedor por medio de SPEED o de cualquier otro protocolo que garantice el intercambio equitativo de bienes.

A partir de disponer de protocolos seguros de pago, podemos pensar en el diseño de frameworks de pago que ofrezcan una serie de interfaces comunes que generen confianza en el usuario y que le permitan elegir entre distintos protocolos a la hora de realizar el pago.

Los distintos frameworks que hemos diseñado son los que aparecen en la parte superior de la Figura 1.3. Estos frameworks están diseñados para principales tipos de productos y servicios electrónicos y buscan soportar de forma genérica el uso de cualquier protocolo de pago. De esta forma clientes y vendedores podrán soportar y utilizar distintos protocolos según los escenarios y según los mecanismos comunes a ambos. Estos frameworks permitirán utilizar tanto SPEED como PURSE-COIN como cualquier otro protocolo existente o que pueda aparecer.

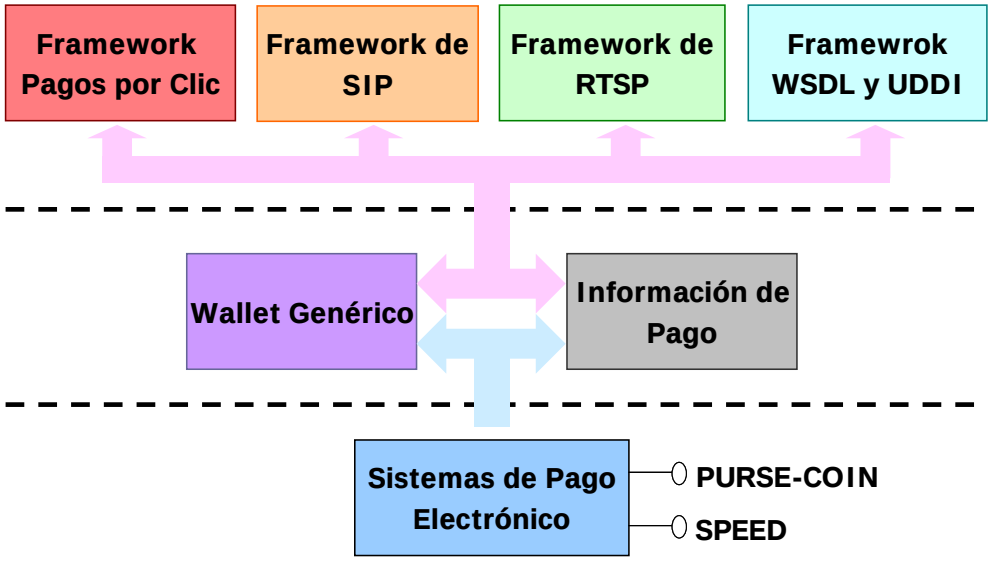


Figura 1.3. Soluciones aportadas.

Para describir la información de pago y para posteriormente utilizar el protocolo de pago elegido, los frameworks utilizan dos elementos que han sido definidos como un elemento intermedio entre los frameworks y los protocolos de pago. Estos elementos son el wallet genérico [255, 256] y la descripción de la información de pago [255, 256] mediante un lenguaje que permita su procesamiento automático como XML.

En un framework es necesario intercambiar la información de pago acerca de los protocolos de pago, marcas financieras (VISA, Mastercard, etc.) y proveedores de servicios de pago soportados (lo que llamamos opciones de pago). Cada opción de pago podría tener asociada un precio distinto ya que los costes de transacción o comisiones de los protocolos pueden variar. Esta información es necesario expresarla en un lenguaje que permita procesarla de forma automática para que los frameworks puedan negociar y soportar que los usuarios puedan elegir el mecanismo a utilizar. En respuesta a esta necesidad

definimos, mediante XML, la información de pago a incorporar en la Web. Esta información también ha sido utilizada en los frameworks para intercambiarla y poder elegir el mecanismo de pago a utilizar en una determinada transacción.

El wallet genérico define una interfaz para poder utilizar cualquier protocolo de pagos. La idea es que los frameworks vean el uso de cualquier protocolo como una caja negra que les abstraiga de las particularidades de cada uno de ellos. En concreto, este wallet ha sido definido como una máquina de estados finitos basándonos en el hecho de que cada protocolo de pagos se podría modelar como el envío y recepción de mensajes. El soporte de los distintos protocolos de pago en los frameworks será a partir de este elemento.

A partir de estos elementos hemos podido definir distintos frameworks de pagos para los principales tipos de productos y/o servicios electrónicos que podemos adquirir en la Web. El hecho de tener diferentes frameworks se debe a que para determinados productos existen protocolos de control estándar que permiten su acceso y que, en caso de que puedan ser extendidos, es conveniente utilizarlos para realizar el proceso de pago a la misma vez que se lleva a cabo el acceso. Este enfoque es adecuado para el uso de micropagos ya que no se necesitan conexiones adicionales ni mensajes adicionales de forma que se pueda llevar a cabo el pago de forma rápida y sin coste adicional.

Los frameworks de pagos diseñados, como decíamos, aparecen en la parte superior de la Figura 1.3. En primer lugar, tenemos el framework de pagos por clic [253, 255, 256]. Este framework está pensado para la adquisición de productos en la Web con la realización de un solo clic. En esta propuesta es dónde se introduce la información de pago a incluir, se define qué es un enlace de pago, se propone un protocolo genérico de pagos orientado a sesión denominado EPP (*Extended Payment Protocol* – Protocolo de Pago Extendido) [253, 256] que soporta la negociación de las opciones de pago y su selección y, finalmente, los wallets que facilitará el uso de distintos protocolos de pago.

A continuación, hemos realizado la definición de un framework para aquellos servicios Web que se quieren ofrecer bajo pago. Para este framework nos basaremos en dos elementos. Uno de ellos acaba de ser mencionado, el protocolo de pagos genérico (EPP) y el otro elemento es PA-WSDL (*Payment Annotations for WSDL* – Anotaciones de pago para WSDL), que es nuestra propuesta para la anotación de servicios Web de pago tanto en WSDL como en UDDI. Esta propuesta facilitará la búsqueda, descubrimiento y negociación de las distintas opciones de pago y de los precios asociados al uso de distintos servicios. Una vez realizado este proceso, el pago se llevará a cabo utilizando el protocolo EPP con el protocolo de pago negociado.

El siguiente framework es el framework de pagos para SIP (*Session Initiation Protocol* – Protocolo de Inicio de Sesión). SIP es un protocolo que permite el establecimiento de sesiones multimedia y que puede ser utilizado para establecer llamadas de Voz sobre IP (VoIP – Voice over IP), sesiones de mensajería instantánea, etc. Este framework se basa en la extensión de los mensajes de SIP para el envío de los mensajes de pago. Para la descripción de la información de pago y para el uso de los distintos protocolos de pago se utilizarán los elementos de las anteriores capas. De este framework también podemos destacar que a la misma vez que permite negociar y elegir las opciones de pago, también permite negociar las características de la sesión (codecs, calidad, etc). Para poder realizar

esta negociación tanto en SIP como en RTSP ha sido necesario extender el protocolo SDP que se utiliza para describir este tipo de sesiones.

Finalmente, nuestra última propuesta es el framework de RTSP, que está orientado al pago de contenidos que se distribuyen bajo demanda. Para éste también se han seguido las mismas ideas que se plantearon para SIP.

1.7. Organización de la tesis

El trabajo realizado en esta tesis doctoral se presentará a lo largo de seis capítulos, en los que se presentarán las distintas soluciones aportadas y que aparecen de forma esquemática en la Figura 1.3.

En este primer capítulo hemos presentado el contexto relacionado con los sistemas y frameworks de pagos para la adquisición de productos y/o servicios electrónicos en entornos B2C en las secciones anteriores (Secciones 1.1 – 1.4). Una vez presentado el contexto hemos presentado, en la Sección 1.5, los objetivos que pretendemos cubrir en esta tesis doctoral. Para la consecución de estos objetivos, en la Sección 1.6, hemos introducido las aportaciones realizadas para satisfaces dichos objetivos. A continuación, en la siguiente sección presentaremos las publicaciones relacionadas con este trabajo y sus contribuciones.

En el Capítulo 2 presentamos un análisis detallado del contexto actual relacionado con los sistemas y frameworks de pago que nos ha llevado a la realización de las propuestas presentadas en esta tesis doctoral. En el capítulo, en primer lugar se introducen los sistemas de pago, se analizan las principales características que deberían satisfacer y se describen brevemente las principales propuestas de pago aparecidos para realizar un análisis comparativo. A continuación, se presentarán las tarjetas inteligentes como un elemento importante de cara a incrementar la seguridad y movilidad de los sistemas de pago. Una vez presentadas las características de las tarjetas inteligentes pasaremos a analizar los principales estándares de pago basados en estas tarjetas. Finalmente, en dicho capítulo se introducen los frameworks de pagos, se describen las características que éstos deberían satisfacer y se realiza un análisis de las principales propuestas aparecidas.

En respuesta a las necesidades planteados en el Capítulo 2 acerca de los sistemas de pago, el Capítulo 3 introduce dos propuestas de pago basadas en monedero electrónico: SPEED y PURSE-COIN. Una vez presentadas estas dos propuestas que permiten mejorar la seguridad de los sistemas de pagos pasamos a abordar el problema de facilitar al usuario la elección y el uso de distintos protocolos de pago de forma que éste se realice por medio de una interfaz conocida. Así, en el Capítulo 4 presentamos un framework de pagos que permite realizar la compra de productos en la Web con un solo clic. Como parte de este framework se definen una serie de componentes como los módulos a incorporar al navegador y al servidor Web, un protocolo de pagos genérico orientado a sesión y que se denomina EPP, la definición de un wallet genérico que permite soportar cualquier protocolo de pago y, finalmente, una serie de elementos XML que nos permitirán definir la información de pago a insertar en las páginas Web y a intercambiar a la hora de elegir las opciones de pago a utilizar para efectuar un pago. A partir de estos elementos y las mismas ideas se ha diseñado un framework de pagos para servicios Web que también se presenta en el mencionado capítulo.

Basado en las ideas y en los elementos definidos en el Capítulo 4, a continuación, en el Capítulo 5 presentamos dos frameworks de pagos que nos permiten la adquisición de otros tipos de productos y servicios electrónicos multimedia. En concreto, los productos y/o servicios son: servicios basados en sesiones multimedia y contenidos que se distribuyen bajo streaming.

Finalmente, en el Capítulo 6 llevamos a cabo un análisis del trabajo presentado a lo largo de los anteriores capítulos. A partir de este análisis expondremos las principales conclusiones a las que hemos llegado después realizar este trabajo. Finalmente, en el capítulo presentamos las principales vías futuras que se derivan del trabajo presentado.

Adicionalmente, se han incorporado una serie de apéndices que nos permiten aportar más detalles acerca de algunas de las propuestas presentadas a lo largo de los capítulos anteriores. En concreto, en el Apéndice A, presentamos el listado de acrónimos utilizados a lo largo de este documento. En el Apéndice B presentamos la especificación HSPSL (High Level Protocol Specification Language – Lenguaje de Especificación de Protocolos de Alto Nivel) de SPEED. En el Apéndice C especificamos la descripción en ASN.1 de SPEED. A continuación, en los Apéndices C y D incluimos las especificaciones de los procesos de certificación y de carga de PURSE-COIN, respectivamente. Finalmente, en el Apéndice F, describimos en mayor detalle todos los elementos XML que son necesarios para la descripción de la información de pago.

1.8. Publicaciones relacionadas

El trabajo de investigación que se presenta en esta tesis ha sido publicado en diversas revistas y congresos nacionales e internacionales. A continuación, presentamos una relación de las principales publicaciones junto con un breve resumen de la aportación realizada.

- A. Ruiz-Martínez, G. Martínez, Ó. Cánovas y A. Gómez-Skarmeta, "SPEED Protocol: Smartcard-Based Payment with Encrypted Electronic Delivery". En actas de *Information Security Conference'01*, Springer, pp. 446-461, 2001.

  Esta publicación [249] presenta el protocolo de pagos SPEED. Este protocolo permite la adquisición de bienes electrónicos por medio de un monedero electrónico. El protocolo además ofrece otras características como la negociación del precio del producto y el envío del producto cifrado. En esta publicación también se presenta una evaluación del protocolo realizada basada en estándares ASN.1.

- F. J. García, A. F. Gómez-Skarmeta, G. López, R. Marín y A. Ruiz. "PISCIS: E-commerce based on an advanced certification infrastructure and smart cards". Revista Upgrade, Vol. III, nº 6, pp. 16-21. 2002.

  El trabajo desarrollado durante el proyecto PISCIS se presenta en este trabajo [113]. Este proyecto es uno de los escenarios de aplicación donde se ha utilizado el protocolo SPEED.

- A. Ruiz-Martínez, L. Baño, C. I. Marín, Ó. Cánovas, A. F. Gómez. "A SECURe infrastructure for negotiation and purchase of Intellectual Property Rights (SECURingIPR)". En actas de *IASTED International Conference on*

Communication, Network, and Information Security (CNIS'03), ACTA Press, pp. 277-282. 2003.

Con el fin de realizar la negociación y adquisición de derechos de propiedad intelectual se desarrolló una infraestructura de seguridad. Uno de los componentes fundamentales en esta infraestructura es el protocolo SPEED, del que se realizó una nueva implementación basada en estándares XML. Este es otro de los escenarios definidos para SPEED.

- A. Ruiz-Martínez, Ó. Cánovas, A. F. Gómez-Skarmeta. "Combination of a Smartcard E-Purse and E-Coin to make Electronic Payments on the Internet". Proceedings International Conference on Security and Cryptography (SECRYPT 2006). Setúbal (Portugal). 2006.

  En este trabajo [251] se presentan los primeros pasos hacia una solución de monedero electrónico que facilite la realización de pagos en la Web.

- A. Ruiz-Martínez, Ó. Cánovas, y A. F. Gómez-Skarmeta, "Smartcard-Based e-Coin for Electronic Payments on the (Mobile) Internet". En actas de *The Third International Conference On Signal-Image Technology and Internet-based Systems (SITIS'2007)*, IEEE Computer Society, pp. 361-368. 2007.

  Un nuevo monedero electrónico que hemos llamado PURSE-COIN se presenta en este artículo [252]. Este trabajo es una versión mejorada del trabajo presentado en [251] con el fin mejorar el uso de los monederos electrónicos para llevar a cabo pagos en la red.

- A. Ruiz-Martínez, Ó. Cánovas, y A.F. Gómez-Skarmeta, "Towards a generic per-fee-link framework". En actas de *2nd International Conference on Digital Information Management (ICDIM'07)*, IEEE Computer Society, pp. 37-42. 2007.

  En este trabajo [253] presentamos un framework de pagos por clic. Este framework ha sido diseñado para soportar el uso de distintos protocolos de pago de una forma genérica. Esta propuesta se basa en el diseño de varios componentes genéricos: módulos para los navegadores y servidores Web, un protocolo de pagos genérico (EPP) orientado a sesiones, el diseño de un wallet genérico basado en un máquina de estados finita y en la información de pago a incluir junto con los enlaces.

- A. Ruiz-Martínez, J.A. Sánchez-Laguna, y A.F. Gómez-Skarmeta, "A Payment Framework for Internet Media-on-Demand Services based on RTSP". En actas *4th IEEE Consumer Communications and Networking Conference 2007 (CCNC 2007)*, IEEE Computer Society, pp. 965-970. 2007.

  El objetivo de esta propuesta [259] es facilitar el uso de protocolos de micropagos para el pago de servicios que se acceden bajo demanda.

- A. Ruiz-Martínez, J.A. Sánchez-Laguna, y A.F. Gómez-Skarmeta, "SIP extensions to support (micro)payments". En actas de 21st International Conference on Advanced Networking and Applications (AINA'07), IEEE Computer Society, pp. 289-296. 2007.

El soporte de (micro)pagos en SIP se puede conseguir por medio de una serie de extensiones que hemos propuesto en este artículo [260].

- A. Ruiz-Martínez, J.A. Sánchez-Laguna, y A. F. Gómez-Skarmeta, “Towards a Generic Payment Framework for Internet Media-on-Demand Services Based on RTSP”. En actas de 22st International Conference on Advanced Networking and Applications (AINA'08), IEEE Computer Society, pp. 826-831. 2008.

  Este trabajo [261] es una versión genérica del trabajo presentado en [259] y que permite el uso de cualquier protocolo (no sólo de micropagos) de pago para conseguir el acceso a contenidos que se distribuyen bajo streaming.

- A. Ruiz-Martínez, Ó. Cánovas, y A.F. Gómez-Skarmeta, “Payment Annotations for Web Service Description Language (PA-WSDL)”. En Computer and Information Science, Springer, pp. 65-76. 2008.

  PA-WSDL [254] introduce una serie de anotaciones que permitirán descubrir aquellos servicios Web de pago y las distintas opciones de pago que éstos soportan.

- A. Ruiz-Martínez, Ó. Cánovas, y A.F. Gómez-Skarmeta, ”Design and implementation of a generic per-fee-link framework”. En Internet Research. 2009. (Aceptado).

  Este artículo [256] es una versión extendida del framework de pagos por clic presentado en [253]. En él se describe la implementación realizada y se lleva a cabo una evaluación del framework.

Además de estas publicaciones, también podemos citar como parte de este trabajo de tesis las participaciones en el II Simposio Español de Comercio Electrónico 2003 (SCE 2003), en la X Reunión Española sobre Criptología y Seguridad de la Información (RECSI 2008) así como una contribución en la Revista Novática (2002).

Capítulo 2

Los Sistemas y Frameworks de Pago Electrónico B2C

A lo largo de este capítulo introduciremos los distintos conceptos relacionados con los sistemas de pagos B2C y las características que éstos deben satisfacer. Entre las distintas características que presentaremos, una de las más importantes es la seguridad que el protocolo puede ofrecer. Una vez analizadas las distintas características presentaremos las diferentes propuestas de protocolos de pago que han aparecido, centrándonos especialmente en aquéllas que nos permiten realizar el pago de productos o servicios electrónicos en entornos B2C. Este escenario es especialmente interesante debido a la gran cantidad de contenidos electrónicos que podemos comprar a través de Internet: libros, MP3s, artículos, revistas, vídeos, etc.

Posteriormente, introduciremos las tarjetas inteligentes como un elemento adicional en un sistema de pago, describiendo sus características y qué pueden aportar al sistema de pago. En particular, la tarjeta actuará como un dispositivo que proporcionará un nivel de seguridad adicional a la hora de gestionar la información criptográfica y de pago así como facilitará el uso del dinero electrónico en distintos entornos (movilidad). Además comentaremos las distintas soluciones de pago basadas en tarjeta inteligente. En especial, nos interesarán aquellas soluciones basadas en tarjetas en las que éstas actúan como contenedor y gestor del dinero y que se denominan monederos electrónicos.

Finalmente, presentaremos los frameworks de pago y analizaremos las principales propuestas. El objetivo de estos frameworks es facilitar tanto a clientes, vendedores y desarrolladores de sistemas de pago el uso de distintos protocolos y mecanismos de pago. En el análisis de las principales propuestas presentaremos sus características más destacadas y cuáles son sus deficiencias a la hora de utilizarlos para realizar pagos en la Web. De esta forma tendremos una visión global de la problemática relacionada con los sistemas de pago electrónico B2C así como de sus frameworks. Esta visión global nos permitirá entender los motivos de las propuestas que presentamos en los capítulos siguientes.

2.1. Introducción

En los últimos años, el uso del comercio electrónico no ha crecido tanto como las expectativas marcaban a finales de la década de los 90. Las razones que justifican el fracaso de estas expectativas se pueden encontrar en [118, 249, 272] y son: la falta de confianza del usuario en los sistemas de pago, sistemas complejos de manejar, interfaces no comunes y/o amigables, seguridad de los sistemas, falta de pilotos que muestren al usuario el funcionamiento de los sistemas de comercio, metodologías para analizar y modelar la seguridad de las transacciones de negocio, etc.

De estos aspectos, como mencionan Tsiakis y Sthephanides en [293], uno de los principales es la confianza del usuario en la seguridad del sistema de pago. Los principales problemas que presentan las propuestas más importantes con respecto a la seguridad son que no tienen en cuenta aspectos como la atomicidad, el intercambio equitativo, el no repudio, no están validadas formalmente y utilizan como método de pago las tarjetas de banda magnética (crédito o débito) que son menos seguras que los monederos electrónicos. En este capítulo analizaremos las principales propuestas de pago con respecto a estos aspectos de seguridad.

En este capítulo también veremos como el uso de las tarjetas inteligentes con monedero electrónico permite ofrecer un mayor grado de seguridad al sistema de cara a proteger la información criptográfica y de pago necesaria para los protocolos. Los monederos electrónicos (tarjeta inteligentes que implementan un sistema de pago) además de aportar seguridad, también facilitan la movilidad de nuestro dinero electrónico. Esto los convierte en el dispositivo ideal para llevar el dinero de forma segura y poder utilizarlo en distintos escenarios como pagos en la Web, pagos en máquinas de vending, etc.

Finalmente, abordaremos otro de los aspectos que han conducido a que el uso de los pagos no sea el esperado. En este caso, cómo simplificar el uso de los sistemas de pago ofreciendo interfaces comunes que sean simples para el usuario y que permitan incrementar su confianza en los sistemas de pago. Para este fin se han propuesto lo que se conocen como frameworks de pagos. En este capítulo realizaremos un análisis de estos frameworks: indicaremos las características que deberían soportar y presentaremos las principales propuestas que han aparecido. En este análisis nos centraremos en estudio de las ventajas e inconvenientes de cada uno de ellos a la hora de realizar pagos en la Web.

2.2. Los sistemas de pago en el comercio electrónico B2C

En esta sección, explicaremos los distintos conceptos relacionados con los sistemas de pago electrónico con fin de comprender las distintas propuestas aparecidas hasta la fecha así como las que posteriormente propondremos en el Capítulo 3. Iniciaremos su caracterización comentando sus principales actores participantes y características distintivas, proporcionaremos una clasificación de estos sistemas y, finalmente, analizaremos los protocolos existentes más destacados. Una vez presentados los sistemas de pago, introduciremos las tarjetas inteligentes, describiremos sus principales características y cómo su uso puede incrementar la movilidad y la seguridad de los

mecanismos de pago. En particular, analizaremos los distintos monederos electrónicos que han sido propuestos para realizar pagos.

2.2.1. Caracterización de los sistemas de pago electrónico

En nuestro mundo físico existen distintos mecanismos para realizar el pago por un producto o servicio. Estos mecanismos de pago son: dinero efectivo, cheques, giros o transferencias de dinero y tarjetas de crédito/débito.

El objetivo de los sistemas de pago electrónico es poder realizar estos tipos de pagos u otros futuros que puedan aparecer, a través de una red de telecomunicaciones. De esta forma, obtendríamos las mismas ventajas que las que presentan otras áreas que se han beneficiado del uso de estas redes. Así, podemos realizar pagos a usuarios que se encuentran en lugares remotos, de forma rápida, en cualquier momento, etc.

Básicamente, un sistema de pago lo podríamos definir como "el nombre colectivo para una forma de realizar una transferencia de valor" [166]. Por tanto podríamos decir que consiste en las especificaciones e implementaciones de protocolos, acuerdos contractuales y estructuras de datos necesarios para llevar a cabo tal proceso.

Actores

En todos los sistemas de pago participan un conjunto básico de entidades desempeñando distintos roles [166, 207, 279]. Como mínimo siempre participará un pagador (la mayoría de las veces desempeñando el papel de cliente o consumidor) y un pagado (representando la mayoría de las veces el rol de vendedor o proveedor de servicios).

En algunos casos, la transferencia de valor del pagador al pagado requiere la participación de entidades financieras que ligan los datos intercambiados en el protocolo de pago a valores monetarios. Esta entidad financiera puede ser un banco, un broker, una organización emisora de tarjetas de crédito o una organización que emita y controle otras formas de representación del valor monetario (p. ej., puntos de fidelización). Normalmente las entidades financieras participan en la transacción desempeñado dos roles: como bancos emisores (interactuando con los pagadores) o como bancos gestores (interactuando con los vendedores). Estos roles incluso podrían ser desempeñados por la misma entidad.

Estas tres entidades (pagador, pagado y entidad financiera) son las más habituales en cualquier sistema de comercio electrónico. Adicionalmente, se podrían considerar otras entidades, como puede ser un árbitro o mediador, que participe en la resolución de disputas [166]. Normalmente, la resolución de disputas se lleva a cabo fuera del protocolo de pago ya que podría estar sujeta a distintas políticas de resolución entre usuarios y entidades financieras.

Otras posibles entidades serían, para determinados sistemas de pago, autoridades de registro o certificación, o incluso terceras partes de confianza que proporcionen anonimato o que velen por el intercambio equitativo.

Características distintivas de los sistemas de pago

Los requerimientos o características que deberían satisfacer los sistemas de pago electrónicos aparecen reflejados en distintos trabajos, tales como [2, 24, 179, 202]. Estos requisitos varían entre las distintas propuestas. Como mencionan Oh [206] o Sahut [266], cabe destacar que el éxito de un sistema de pagos electrónico no depende de un solo factor, sino que depende de diversos aspectos de distintas áreas o dimensiones. Así, la valoración de un protocolo no sólo depende de aspectos técnicos sino también de aspectos económicos, sociales, institucionales y de regulación. Dependiendo de los autores o del escenario, se les da más importancia a unos requisitos que a otros. A continuación, pasamos a indicar cuáles son los más interesantes:

- *Seguridad*. Debido a que los sistemas de pago electrónico involucran dinero, se les debe dotar de medidas de seguridad que protejan el intercambio de dinero frente a distintos tipos de ataques que se pueden producir en la red, como por ejemplo la escucha de información o la modificación de mensajes. Los ataques que se pueden producir habitualmente se clasifican como pasivos (en los que el atacante sólo monitoriza el tráfico de la red para obtener información que posteriormente podría utilizar, como pueden ser contraseñas, el número de la tarjeta de crédito, etc.) y activos (en los que el atacante interfiere en el tráfico de la red), por ejemplo, suplantando la identidad de un usuario, reenviando mensajes o impidiendo la comunicación. El nivel de seguridad ofrecido dependerá de varios aspectos: cantidad a pagar, posible nivel de fraude, confianza, etc. El grado de seguridad implica: seguridad en el proceso de depósito o retirada de dinero; datos, la seguridad de las aplicaciones y bases de datos; la seguridad durante las transacciones y los pagos; la seguridad de Internet y del sistema; y el mantenimiento y gestión de la seguridad. De entre estas, las más importantes para compañías y usuarios son la seguridad de la transacción y del pago. Estas propiedades necesitan satisfacer, a su vez, los siguientes requisitos: autenticación, integridad confidencialidad y no repudio.
- *Prevención total del fraude*. En un sistema de pagos, un aspecto fundamental es prevenir los fraudes relacionados con el pago, tales como la creación de dinero falso, el robo de este dinero y, finalmente, el *doble gasto* (que un cliente pueda utilizar la misma moneda electrónica para realizar varios pagos).
- *Escalabilidad*. La infraestructura de pagos debe ser capaz de manejar el incremento de usuarios sin que se note una importante pérdida de rendimiento. Para lograr este objetivo, la infraestructura debe soportar múltiples servidores distribuidos a través de la red y que los protocolos utilizados no degraden el sistema de forma importante cuando el número de usuarios crece.
- *Eficiencia*. El proceso de pago se debe realizar de forma eficiente con el fin de que se puedan soportar el mayor número de transacciones posibles en un momento dado. La eficiencia cobra especial importancia cuando se quiere realizar el pago de contenidos o servicios cuyo precio es una pequeña cantidad sin que se aprecie una degradación del sistema. El coste por transacción de usar esta infraestructura debe

ser lo suficientemente pequeño de forma que sea insignificante, incluso para transacciones de cantidades del orden de céntimos de euro.

- *Confidencialidad y Anonimato*. Se debería asegurar que la información de pago sólo es accesible a aquellos que están autorizados a tener acceso a ésta (confidencialidad). Además, en algunas transacciones, la identidad de los participantes no debería de ser revelada a vendedores y entidades financieras (anonimato). El anonimato puede presentar un riesgo de seguridad que, en determinados tipos de aplicaciones, puede considerarse inaceptable [202]. Por otro lado, el anonimato revocable significa que no se revela la identidad del cliente salvo que éste incurra en un fraude. También se debe evitar la monitorización de los patrones de compra de los usuarios, así como otra información relativa al usuario durante una transacción de pago (confidencialidad). Normalmente, cuando el anonimato es importante, el coste de trazar una transacción debería de sobrepasar el valor de la información que se puede obtener [202]. También hay que tener en cuenta que el uso del anonimato debería ser flexible ya que, por ejemplo, hay países que no lo aceptan como es el caso de los Estados Unidos de América (EEUU), desde el 11 de Septiembre de 2001, puesto que quieren controlar cualquier movimiento de dinero que pueda ser sospecho.
- *No repudio*. El no repudio pretende proporcionar las evidencias necesarias para demostrar que una determinada entidad participó en una transacción [324]. El objetivo de estas evidencias es evitar que, posteriormente, una de las entidades pueda negar su participación cuando realmente sí participó. Para garantizar el no repudio, una de las principales herramientas es el uso de la firma electrónica sobre determinada información relacionada con la transacción. Un análisis más detallado acerca de los distintos elementos necesarios para satisfacer el no repudio se puede encontrar en [104, 324].
- *Intercambio justo o equitativo (Fair exchange)*. Se dice que un intercambio en un protocolo (por ejemplo, el pago con una moneda a cambio del producto) es justo o equitativo si o bien ambas partes reciben lo que esperan del intercambio o por el contrario ninguna de ellas obtiene nada en el intercambio. Esta propiedad es distinta de la anterior ya que, podría darse el caso de que, un protocolo garantice el no repudio y, sin embargo, no sea justo [112, 324].
- *Atomicidad*. Esta propiedad indica que o bien la transacción ocurre completamente o, por el contrario, no se produce ningún efecto. Para Tygar [295, 296], la atomicidad representa la característica que evita la creación o destrucción de dinero durante el intercambio (esta propiedad no representa un intercambio justo). Tygar también define lo que se llama *buena atomicidad* y que representa la atomicidad y el intercambio justo de un bien por una moneda.
- *Confianza*. Las anteriores propiedades, si se satisfacen adecuadamente, generan confianza en el sistema. Aquí entendemos por confianza el grado de confianza en que tanto el dinero como la información personal estarán seguros y que las partes involucradas no actuarán en contra de los intereses del usuario. Un sistema confiable diremos que lleva a cabo los pagos de una forma segura y que el dinero no

será perdido/robado o mal usado. Finalmente, el sistema será confiable si, aún no siendo perfecto, el usuario cree que el sistema se está comportando correctamente. Un análisis sobre la seguridad y confianza en B2C se puede encontrar en [293, 318].

- *Tipo de autorización* o *validación*. Es la capacidad del sistema para llevar a cabo pagos sin la necesidad de conectarse a una autoridad central o a una tercera parte. La autorización puede ser en línea (on-line), fuera de línea (off-line) o semi en línea (semi-online). Estas categorías se analizan en más detalle en el siguiente apartado.
- *Divisibilidad*. Entenderemos por divisibilidad el hecho de que a la hora de utilizar un medio de pago, el valor del dinero electrónico almacenado se pueda dividir en importes más pequeños cuando se efectúa el pago de forma que éste se ajuste a la cantidad a pagar.

Categorización de los sistemas de pago

A día de hoy existe un amplio abanico de sistemas de pago en el comercio electrónico [207, 279]. Esta diversidad está motivada por las distintas características que cada uno de ellos soporta o por el ámbito de aplicación [157].

Entre las características que podemos utilizar para clasificar un sistema de pago podemos mencionar: modelo de negocio (momento en el que se realiza el pago), cantidad a pagar, tipo de validación del pago, el tipo de dispositivo y según cómo se transfiere el dinero. A continuación, presentaremos cada una de estas clasificaciones.

Según el modelo de negocio (momento en que se realiza el pago)

Los mecanismos de pago ofrecidos a los clientes pueden soportar distintos modelos de negocio. En este caso, estos modelos de negocio vienen determinados por el momento en el que se lleva a cabo la liquidación del pago al cliente. Estos modelos son: *prepago* (pagar antes), *débito* (pagar ahora) y *crédito* (pagar después).

En el *modelo de prepago*, el cliente retira dinero electrónico de una determinada entidad (banco, broker o vendedor). Este dinero se carga en la cuenta del cliente. El dinero electrónico retirado se almacena como un valor en una tarjeta inteligente (p. ej. tarjetas telefónicas de prepago), en cuentas de cliente específicas o como monedas electrónicas. Este dinero posteriormente podrá ser utilizado por el cliente para realizar pagos.

En el *modelo de débito*, el cobro al cliente se realiza justo en el momento en el que la transacción tiene lugar, como es el caso de las tarjetas bancarias de débito.

Finalmente, en el *modelo basado en crédito*, cuando un cliente realiza una transacción, el importe de ésta no se le cobra en el momento del pago. El cobro del pago al cliente se realiza al final de un determinado periodo de tiempo establecido entre el cliente y la entidad que le concede el crédito (normalmente una entidad financiera). En este caso podemos encontrar las tarjetas bancarias de crédito.

Según el importe del pago

De acuerdo al importe que se puede pagar con un protocolo [2], los sistemas de pago los podemos clasificar como protocolos de *micropagos*, de *pequeños pagos* y de *(macro)pagos*. Los límites establecidos varían según el autor [2, 157, 237, 293, 326].

En nuestro caso, consideraremos que en un protocolo de *micropagos* las cantidades a pagar no son superiores a un Euro. En estos protocolos prima la eficiencia frente a la seguridad del sistema. Ejemplos de protocolos de micropagos son: cheques de viaje [171], e-coupons [215], Micromint [238], Millicent [117], NetCents [228], NetPay [67], Payword [238], Peppercoin [237], etc. Un análisis de estos protocolos se puede encontrar en [132, 168, 192, 207, 268, 279, 291].

En los *pequeños pagos* el importe de las transacciones se encuentra entre un Euro y cinco Euros. Algunos ejemplos podrían ser NetBill [63], LITESET [122], CONSEPP [169] y LITESET/A++ [308].

Finalmente, los protocolos de *(macro)pagos*, que manejan transacciones superiores a cinco Euros, se centran en proporcionar seguridad en la transacción debido a los altos importes que pueden llegar a manejar. Ejemplos de estos protocolos son SET [191, 274], SET/A+ [316], 3D-SET [140, 300], la propuesta de Garner et al. [116], etc.

Según el tipo de validación

Atendiendo al tipo de validación que se realiza sobre el pago, podemos clasificar a los sistemas de pagos en sistema *off-line* (fuera de línea), sistemas *on-line* (en línea) y sistemas *semi-online*.

En los *sistemas off-line*, durante el proceso de pago, no se requiere la validación del pago por parte de una tercera parte, como podría ser un banco o emisor de tarjetas de crédito. En cambio, en los *sistemas on-line* sí se requiere este tipo de validación por cada pago. Finalmente, los sistemas *semi-online* requieren cierta interacción con una tercera parte pero no para cada pago.

La principal ventaja de los sistemas off-line es que el proceso de pago puede llegar a realizarse de forma más rápida, ya que no se requiere la participación de una tercera entidad; sin embargo, el riesgo puede ser mayor ya que podrían producirse problemas de doble gasto. Por otro lado, este tipo de riesgos son fácilmente evitables en los sistemas on-line; sin embargo, este tipo de sistemas supone una mayor sobrecarga en las comunicaciones debido a la participación de esta entidad. Finalmente, los sistemas semi-online intentan combinar las ventajas de ambos de forma que la mayoría de las operaciones se realicen de forma off-line y sólo en aquellos casos en que se determine que existe un riesgo importante es cuando se realizaría la validación on-line.

Según el dispositivo

Como ya se avanzó anteriormente, una de las características que permite definir a un sistema de pago es el tipo de dispositivo utilizado para realizar el pago. Dependiendo del tipo de dispositivo, podríamos estar hablando de *sistemas de pago* y *sistemas de pago móvil*.

En los sistemas de pago móvil se acentúa el hecho de disponer de un dispositivo con unas características especiales en cuanto a comunicación, tamaño, etc. Este tipo de dispositivos nos permite utilizar distintos tipos de redes inalámbricas para comunicarnos y, por lo tanto, podemos utilizarlos en distintas ubicaciones. A diferencia de los que simplemente denominaremos sistemas de pago, en los que se supone que nos conectamos desde un dispositivo que no dispone de capacidades de movilidad en la red, sino que está conectado a una red cableada.

Según cómo se transfiere el dinero

Otra categorización diferente es la propuesta por Abrazhevich en [2], en la que los sistemas de pago se pueden clasifican atendiendo a cómo se transfiere el dinero. Así, los sistemas de pago los clasifica en (ver Figura 2.1): *sistemas de dinero electrónico o digital* y *sistemas de crédito/débito*. Wayner [310], en cambio, los denomina basados en token y basados en cuenta, respectivamente.

En la Figura 2.2a) se muestra el flujo que tiene lugar en un sistema basado en dinero digital. En este tipo de sistema, generalmente, en primer lugar (paso 1), el pagador retira un conjunto de monedas electrónicas del emisor. Estas monedas le permitirán realizar un pago al pagador (paso 2) que, para obtener el dinero de las monedas electrónicas, las depositará en su banco (paso 3). Posteriormente, se producirá la liquidación entre el emisor y el banco del vendedor.

En la Figura 2.2b) podemos observar cómo se realiza la transferencia de valor en un sistema basado en cuenta (similar a los sistemas de cheque tradicional). El pagador inicia el proceso con el pagado enviándole una autorización de pago (paso 1). A continuación el pagado la presenta a su banco (el banco gestor) que, a su vez, la remite al banco del cliente (banco emisor) para que apruebe la transacción. Si la transacción se aprueba, el cliente recibe una confirmación y, posteriormente, se realiza el proceso de liquidación entre bancos.

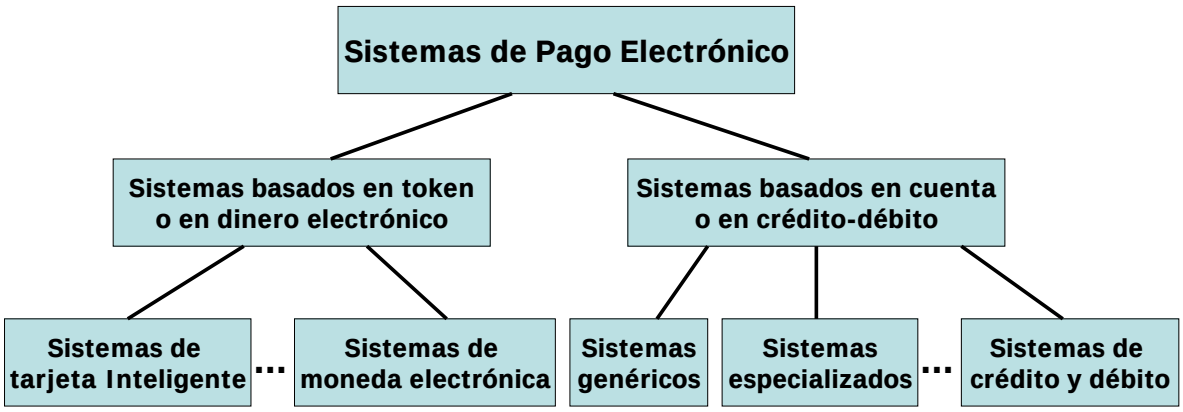


Figura 2.1. Clasificación de los sistemas de pago según Abrazhevich.

Como ya mencionábamos, el dinero electrónico pretende ser el equivalente al dinero tradicional representado como monedas y billetes. En este caso, el dinero se representa

como una serie de bytes (o token). El pago, en general, se realiza transfiriendo el token de una entidad a otra. Por otro lado, los sistemas de crédito y débito representan el uso de cuentas bancarias y/o tarjetas de crédito. En este caso, el dinero se representa por un determinado valor numérico asociado a un número de cuenta o de la tarjeta de crédito asociado. Así, el pago consiste en el decremento del valor de una cuenta (la del cliente) y el incremento de otra cuenta (la del vendedor).

Continuando con la clasificación descrita por Abrazhevich, los sistemas de dinero electrónico los podríamos clasificar en (ver Figura 2.1): *sistemas basados en tarjeta inteligente o monedero electrónico* y *sistemas basados en moneda electrónica*. Los sistemas basados en monedero electrónico representan el dinero como un contador dentro de una tarjeta electrónica. Ejemplos de este tipo son: WG10 [44] (que lo utilizaremos como parte del protocolo SPEED que será presentado en el Capítulo 3) y CEPS [45]. En cambio, los sistemas basados en moneda electrónica, utilizan una serie de bytes que representan el valor equivalente a una moneda. ECash [49], Millicent [117], NetCash [188], Netpay [67], Paycash [218, 219] y Payword [238] son ejemplos de este tipo de sistemas. Posteriormente, en el Capítulo 3 veremos que la propuesta denominada PURSE-COIN combina ambos sistemas a la hora de realizar pagos.

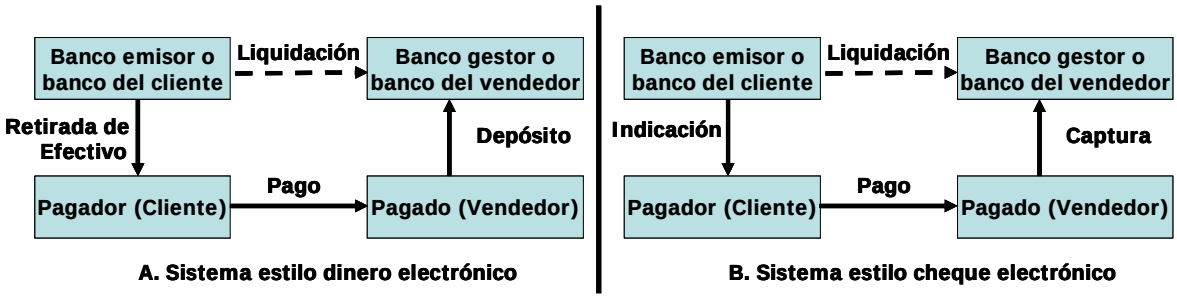


Figura 2.2. Transferencia de dinero según el sistema de pago.

Los sistemas de moneda electrónica se propone clasificarlos en *sistema de moneda electrónica genérica* y *sistemas de moneda electrónica específica*. Esta clasificación la realizamos porque queremos distinguir claramente las ventajas de cada una de ellas ya que, posteriormente, PURSE-COIN, combina las ventajas de ambos tipos. En los sistemas de moneda genérica, las monedas que recibe el cliente puede utilizarlas con cualquier vendedor. Estas soluciones dan más flexibilidad al cliente. Sin embargo, normalmente, se requiere que el vendedor establezca una conexión con el banco o con otro vendedor para comprobar que la moneda no se ha gastado previamente. Por otro lado, los sistemas de moneda específica, sólo permiten que la moneda se utilice con un vendedor particular. Esto limita el uso de las monedas. Sin embargo, permite realizar un proceso de verificación de la moneda sin que sea necesaria la participación de un tercero.

Los sistemas basados en cuenta los podemos dividir en (ver Figura 2.1): sistemas *basados en crédito/débito*, *sistemas especializados* y *sistemas genéricos*. Los sistemas de crédito/débito se basan en el uso de una tarjeta de crédito/débito para realizar el pago. Los sistemas especializados se basan en determinadas características de un sistema para

realizar pagos, por ejemplo, los pagos por medio de correo electrónico. Finalmente, los sistemas genéricos permiten realizar el pago basado en una cuenta genérica dentro de un sistema en particular. Posteriormente, el cargo dentro de esta cuenta genérica se cobrará mediante una cuenta bancaria o mediante la tarjeta de crédito/débito. Un ejemplo de este sistema podría ser NetBill [63] o PayPal [217].

2.2.2. Principales sistemas y protocolos de pago electrónico

En esta sección describiremos algunos de los protocolos de pago más significativos aparecidos hasta la fecha para entornos B2C. Estos protocolos serán analizados de acuerdo a las características mencionadas en el apartado de características distintivas de los sistemas de pago (ver Sección 2.2.1). Un amplio estudio de los distintos protocolos de pagos se puede encontrar en [207, 279].

SSL/TLS

SSL (Secure Socket Layer) [107, 133, 236] o su versión actualizada y estandarizada TLS (Transport Layer Security) [73, 236] no es en realidad un protocolo de pagos, sino que se trata de un protocolo para proteger la información que se intercambia a nivel de transporte y que es independiente del protocolo de la capa de aplicación. Este protocolo garantiza la integridad, autenticidad y la confidencialidad de la información intercambiada entre dos entidades. Una de estas entidades actúa como servidor (normalmente el servidor Web de una entidad como puede ser un vendedor, un ISP, etc.). La otra entidad actúa como cliente (normalmente el navegador Web de un cliente). En este intercambio de información, por defecto, sólo se autentica el servidor por medio de certificados digitales. Opcionalmente, y a petición del servidor, se podría requerir la autenticación del cliente por medio de un certificado digital.

A día de hoy, este protocolo es el que más se utiliza para realizar pagos en Internet [277]. El proceso de pago se inicia una vez que el cliente ha realizado el proceso de selección de los productos. Este proceso comienza con el establecimiento de un canal seguro con SSL/TLS con el vendedor. Una vez confirmada la información de los productos así como determinada información que pueda ser necesaria para la entrega del producto se procede a solicitar la información de la tarjeta necesaria para realizar el pago. La información bancaria se reenvía a la entidad financiera que aprueba o deniega la transacción. Si la transacción se aprueba, el vendedor es notificado y pasará a suministrar el producto o servicio al cliente. En una transacción de pago con SSL hay dos formas de enviar la información bancaria del cliente a la entidad financiera. En la primera de ellas es el vendedor quien se la envía a la entidad financiera. En la segunda de ellas, cuando se inicia el pago, el vendedor redirige al cliente a la entidad financiera para realizar el pago, de forma que la información de la tarjeta de crédito sólo la conoce la entidad financiera. Una vez efectuado el pago, la entidad financiera redirige al cliente al vendedor.

A continuación mencionaremos las principales ventajas y desventajas de este protocolo. Algunas de ellas aparecen en [104] y [236]. La principal ventaja de este esquema es que es sencillo de implementar ya que todos los servidores Web y todos los navegadores actuales

lo soportan. Además, la presencia de este protocolo es transparente tanto al software del comercio del vendedor como al del cliente, lo que facilita la integración de su uso con las aplicaciones del vendedor. Para los vendedores no hay ningún coste de integración de SSL/TLS en sus sistemas, salvo el de instalar un certificado. Para los clientes el único requisito es el soporte de SSL/TLS, que ya viene soportado por el navegador.

Este protocolo también garantiza que la integridad y la confidencialidad de la información intercambiada entre cliente y servidor así como la autenticación del servidor (y opcionalmente la del cliente). También facilita la movilidad de los clientes ya que los clientes pueden realizar sus transacciones en cualquier ordenador que disponga de un navegador Web. Otra ventaja adicional es su baja complejidad, de forma que el protocolo supone un impacto mínimo en el tiempo de la transacción así como puede soportar un número elevado de conexiones simultáneas (escalabilidad) [78]. Finalmente, se puede especificar la cantidad exacta por la que se desea realizar el pago.

Cabe mencionar que el protocolo ha sido ampliamente estudiado [196, 205, 216, 304, 323] y que sólo se han encontrado fallos menores y algunos ataques activos importantes pero que puede ser corregidos fácilmente como se propone en las propuestas que acabamos de reseñar. Por estas razones, este método es altamente aceptado y posee una amplia base de clientes que poseen una tarjeta de crédito/débito.

El principal problema de este esquema es que no garantiza el no repudio de la transacción ni el intercambio justo ni la atomicidad. De forma que el cliente podría realizar el pago y en cambio podría no recibir el producto solicitado a cambio. Además, el cliente no tendría evidencias para probar que no recibió el producto.

Como en la mayoría de los casos, no se requiere autenticación de cliente (a menudo los usuarios no tienen un certificado, ya que no son obligatorios y, por tanto, raramente usados), el vendedor no está seguro de si realmente está autenticando al usuario o no. Esto es especialmente perjudicial para los vendedores ya que ellos son los responsables del coste de la devolución de las compras donde “la tarjeta no está presente” (“card not present” – se refiere a las transacciones que se realizan en remoto) [14, 15]. Incluso aunque el usuario poseyera un certificado, éste no tiene por qué estar asociado con su tarjeta de crédito.

En el caso de que sea el vendedor quien envía la información al banco gestor, existen problemas adicionales. Así, en este modo el vendedor conoce la información bancaria relacionada con el usuario (su número de tarjeta de crédito y la información asociada para el pago). Por tanto, no garantiza la privacidad. Tampoco impide que el vendedor reciba un número de tarjeta fraudulento. Además, la información recibida podría ser utilizada por el vendedor para realizar cobros adicionales al usuario. Este almacenamiento también puede constituir un problema si el vendedor es atacado y el atacante consigue acceder a su base de datos, donde tendría disponible la información de las tarjetas de crédito de los usuarios.

SET

El protocolo SET (Secure Electronic Transaction) [4, 77, 191, 274] es una propuesta de Visa y Mastercard que data del año 1997 y que tiene por objetivo proteger la privacidad de la información de pago y asegurar su autenticidad [21, 77, 191] para evitar los problemas que presentaba SSL/TLS a la hora de realizar pagos mediante tarjeta de crédito. A diferencia de SSL/TLS, SET sí es un protocolo de (macro)pagos basado en tarjeta de crédito.

Una transacción en SET incluye los siguientes participantes: el cliente, el vendedor, el emisor de la tarjeta de crédito del cliente, el banco del vendedor y una pasarela de pagos. Cada una de las partes implicada en la transacción es autenticada mediante el uso de certificados digitales que están emitidos por una autoridad de certificación que específicamente emite certificados para este protocolo.

El proceso de pago se explica a continuación (ver Figura 2.3). En primer lugar, tiene lugar la fase de inicialización con el fin de que el cliente pueda obtener los certificados del vendedor y de la pasarela de pagos.

A continuación, se inicia el pago de forma que el cliente envía al vendedor tanto la información de compra (el producto o servicio solicitado) como los datos necesarios para realizar el pago (número de la tarjeta de crédito, fecha de expiración, etc.). Sin embargo, el vendedor no podrá acceder a la información de la tarjeta de crédito del usuario, ya que esta información sólo la obtendrá la pasarela de pagos.

Una vez comprobada la solicitud, tiene lugar la fase de autorización en la que el vendedor contacta con la pasarela de pago para saber si se autoriza el pago o no y si el vendedor puede proceder a la entrega del producto o servicio solicitado.

Cuando el vendedor recibe la autorización, pasa a enviar al cliente la confirmación del pago que, además, podría actuar a modo de recibo para el cliente.

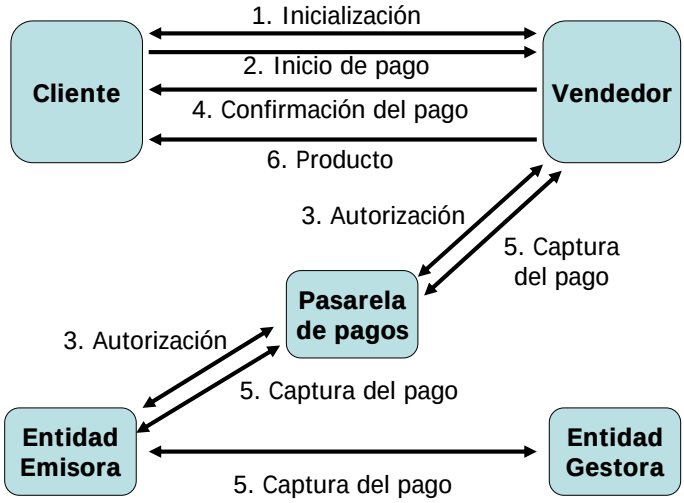


Figura 2.3. Modelo de pago en SET.

Posteriormente, el vendedor ejecutará la fase de captura/liquidación del pago en la que el vendedor hace efectivos todos los pagos que ha recibido durante un período de tiempo. Una vez recibido el cobro, el vendedor puede suministrar el producto al cliente.

Como puntos destacados de esta propuesta podemos señalar que satisface los requisitos de seguridad tales como: confidencialidad, autenticación, integridad y resistencia a ataques de reenvío, aunque no proporciona mecanismos de negociación segura. Estas propiedades han sido verificadas por una amplia colección de pruebas de seguridad basadas en métodos formales [20, 22, 23, 51, 131, 176, 262]. Además, el protocolo garantiza que cada parte sólo tenga visible la información que realmente necesita conocer, de forma que el vendedor no conocerá la información bancaria del cliente y la pasarela de pagos no conocerá la información relacionada con la compra del producto o servicio. Finalmente, supone una ventaja para los vendedores, ya que se reducen las posibilidades de anulación de las transacciones ya realizadas por el cliente.

A pesar de estas ventajas, SET no ha conseguido obtener el grado de éxito que se esperaba y no ha obtenido una adopción significativa, ya que no ha alcanzado una amplia aceptabilidad. Las razones de este fracaso son varias. En primer lugar, comparado con SSL/TLS, SET es más complejo ya que requiere que el cliente tenga un certificado digital válido específico para SET. A día de hoy este es un problema menor, ya que el uso de los certificados está más extendido. Sin embargo, en el momento de su aparición esta suposición no era factible. Relacionado con este problema, otro inconveniente adicional es que el usuario sólo podía realizar compras en el ordenador en el que tuviera instalado este certificado o, alternativamente, tenía que transportarlo mediante dispositivos de almacenamiento (como un disco) e instalarlo en un nuevo ordenador, con el riesgo de seguridad que suponía instalar el certificado y la clave en un ordenador que no estaba bajo el control del usuario. Esta situación, gracias al cada vez más extendido uso de las tarjetas inteligentes, ya no sería tan problemática.

Desde el punto de vista del vendedor requería que el vendedor tuviera cuenta bancaria con aquellas entidades bancarias que eran capaces de operar con SET.

Desde el punto de las entidades bancarias requería que ellos mismos desarrollaran las pasarelas de pagos o que confiaran su desarrollo a un tercero de confianza.

El protocolo, desde el punto de vista criptográfico, supone una mayor sobrecarga que SSL/TLS. Sin embargo, tal y como se menciona en [4], este coste se podía considerar asumible por las distintas entidades.

SET aunque es seguro para los requisitos anteriormente mencionados, sin embargo, no garantiza correctamente el no repudio, ya que el cliente en una transacción en SET no obtiene un recibo seguro del pago [131]. Existe también un ataque por medio del cual, un cliente deshonesto podría comprar productos a un vendedor honesto (con la ayuda de otro vendedor) por el que el cliente no paga [31]. Finalmente, algunos autores como Bella et al. [23] y Cheng et al. [51], han encontrado algunas pequeñas deficiencias que serían fácilmente subsanables como ellos mismos proponen.

Modelos de pago basados en tres dominios (3D): 3D SET y 3-D Secure

En esta sección explicaremos el modelo de pago en tres dominios (3D) sobre el que se basan las propuestas 3D SET [140, 300] y 3-D Secure [301]. Estos dos modelos de pago se analizarán de forma conjunta debido a las similitudes que presentan.

3D SET es una iniciativa Visa Internacional introducida en 1999 que pretende mejorar algunas de las limitaciones de SET como, por ejemplo, que el cliente tenga que instalar software específico para ejecutar una transacción de pago o que tenga que poseer un certificado.

El modelo de pago 3-D Secure fue desarrollado por Visa en 2000 con el fin de mejorar la seguridad de las transacciones en Internet. En concreto, la autenticación de la información bancaria que el cliente envía al vendedor mediante SSL/TLS.

Los participantes básicos en este modelo 3D son: cliente, vendedor, banco del cliente o emisor, banco del vendedor o banco gestor y una pasarela de pagos. Este modelo de pago define tres dominios: el dominio del banco del cliente, el dominio del banco del vendedor y el dominio de interoperabilidad.

El dominio del banco del cliente refleja la relación entre el cliente y su banco. El dominio del banco del vendedor, de forma similar al anterior, cubre la relación entre el vendedor y su banco. Finalmente, el dominio de interoperabilidad soporta la relación entre el banco del cliente y el banco del vendedor. En este modelo basado en tres dominios la autenticación entre las partes se lleva a cabo dentro de cada dominio. De esta forma, el banco de cliente y el banco del vendedor deciden, dentro de su dominio, la forma en que realizarán la autenticación del cliente y del vendedor, respectivamente. Los mecanismos pueden ser desde un usuario/contraseña a una firma digital.

En ambas propuestas, de forma genérica el proceso de pago sería el siguiente. El cliente y el vendedor intercambian la información necesaria para efectuar la transacción. A continuación, el vendedor redirige al cliente al banco de éste último para que se autentique frente a él. Una vez que el cliente se autentica con su banco, éste es redirigido de nuevo al vendedor. A partir de la información que le proporciona el cliente, el vendedor solicita a su banco una autorización de pago. A continuación, el banco del vendedor solicita una autorización de pago al banco del cliente. Una vez que el banco del vendedor ha obtenido la autorización, el vendedor pasa a emitir un recibo de la transacción y a realizar la entrega del producto o servicio.

La principal ventaja de ambas propuestas para los vendedores es que si soportan este modelo no tendrán que hacerse cargo de las devoluciones de productos ocasionadas por la anulación de peticiones de los clientes, ya que se puede probar la participación del cliente en la transacción. En ambos esquemas, a diferencia de SSL/TLS, la información bancaria del cliente está autorizada por su banco. Otra ventaja adicional es la flexibilidad del modelo a la hora de permitir distintos mecanismos de autenticación que se adapten a las diversas posibilidades de los bancos emisores de tarjetas. Para el cliente, la principal ventaja la representa el hecho de que la transacción se realiza de forma segura por un servidor de su banco.

Los principales problemas que presentan ambas propuestas son los siguientes. En primer lugar, la seguridad del sistema depende del sistema de autenticación utilizada para autenticar al usuario. En este modelo se requiere que el cliente deposite toda su confianza en el banco. Dependiendo del método utilizado para autenticar al usuario se puede garantizar o no el no repudio. El usuario al no tener el control, sería posible que el banco generase transacciones de pago sin que el cliente realmente las haya realizado. Esta propuesta tampoco garantiza el intercambio justo ya que un cliente podría realizar el pago y, por el contrario, no recibir el producto por el que pagó. Finalmente, se produce una mayor participación y, por tanto, sobrecarga en los servidores de las entidades bancarias ya que su participación es imprescindible en el proceso de pago. Finalmente, ambos esquemas al estar basados en la redirección HTTP son susceptibles a ataques man-in-the-middle, de forma que el usuario podría ser redirigido a falsos servidores [214].

NetBill

NetBill [63, 153] es un sistema de pago electrónico desarrollado en 1995 por la Universidad de Carnegie Mellon en asociación con Visa y el banco Mellon y orientado a la compra-venta de información de bajo precio.

En este sistema participan los clientes, las entidades comerciales, y el servidor NetBill, quien se encarga de mantener cuentas de los clientes y los vendedores, pudiendo estas cuentas estar asociadas a cuentas tradicionales en entidades financieras. El importe de las compras realizadas por los clientes se carga en su cuenta NetBill. Posteriormente, el servidor NetBill abona el importe pagado a los vendedores.

La Figura 2.4 muestra el proceso de pago en NetBill. Este protocolo se analizará en más detalle debido a las similitudes que presenta con una de las propuestas que introducimos en el Capítulo 3 (SPEED). El protocolo se inicia cuando un cliente solicita el precio de cierta información o contenido en un servidor mediante el envío de una oferta formal (paso 1). A continuación, el cliente y el vendedor pueden negociar el precio del producto ya que el protocolo proporciona mecanismos para regatear el precio base o para negociar descuentos por volumen de compra, etc.

El vendedor, tras recibir la solicitud de un cliente, decide un precio para su oferta y la envía al cliente (paso 2). Si éste está de acuerdo con el precio, envía al vendedor una solicitud de pago (paso 3). Una vez recibida la solicitud, el vendedor envía al cliente el producto cifrado con una clave de un solo uso (paso 4).

Para proceder con el pago y obtener la clave con la que se cifró el producto, el cliente envía una orden de pago electrónica (EPO) firmada al vendedor (paso 5). En cualquier momento, antes de enviar la EPO, el cliente puede cancelar la transacción. El envío de la EPO marca el "punto de no retorno" para el cliente. Al recibir la EPO, el vendedor la reenvía al servidor NetBill (paso 6).

El servidor NetBill verifica que el precio, el resumen digital y otra información relacionada con el pago son correctos y carga en la cuenta del cliente la cantidad correspondiente. A continuación, almacena los datos de la transacción y guarda una copia de la clave de un solo uso. Una vez realizado este almacenamiento, el servidor NetBill

envía al vendedor un mensaje firmado con la aprobación o denegación de la transacción (paso 7). Este mensaje firmado junto con la clave que descifra la mercancía es lo que recibe el cliente del vendedor (paso 8).

Todas las transacciones en las que se compra información son atómicas. En una situación ideal donde sólo se dan los ocho pasos descritos, el vendedor contacta únicamente una vez con el servidor NetBill. En caso de desavenencias, un cliente puede contactar con el servidor NetBill directamente.

La principal ventaja de este esquema es que se garantiza que el cliente sólo paga por la información que ha solicitado, el vendedor cobra por la información vendida y además, en caso de que alguno intente engañar al otro, se disponen de las pruebas suficientes para poder probar quién es el que está defraudando. Por tanto el protocolo garantiza la atomicidad, el no repudio y el intercambio equitativo. Sin embargo, existe un ataque a la propiedad de buena atomicidad. Así es posible un ataque en el que un cliente que paga por un producto a un vendedor fraudulento podría no recibir el producto correctamente y no tendrá pruebas que lo acredite. Este ataque ha sido probado formalmente por Ogata y Futatsugi [205]. Estos mismos autores proponen una solución basada en que el servidor NetBill reciba también el producto. Sin embargo, esta propuesta puede suponer problemas de escalabilidad para el servidor NetBill.

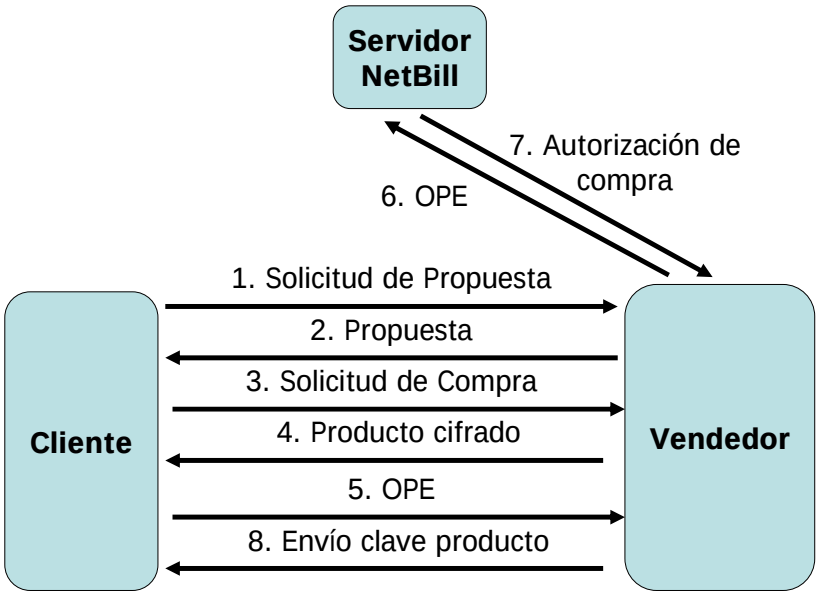


Figura 2.4. Proceso de pago en NetBill.

El protocolo además garantiza que la información intercambiada entre las distintas partes no será observada por terceras personas, ya que los mensajes van cifrados con una clave que sólo conocen las partes involucradas en la transacción. Es más, el servidor de NetBill, salvo en caso de que se produzca un engaño y las distintas partes le tengan que presentar la información de la compra para ver quién engaña a quién, tampoco conocerá los productos que compra el cliente al vendedor, conociendo únicamente que un cliente ha

comprado algo a un vendedor por una determinada cantidad. Por tanto la confidencialidad está garantizada.

Es de destacar también, que el protocolo ofrece una solución integral de seguridad tanto para la negociación del precio del producto como para la compra. No hay que olvidar que la negociación también es una etapa importante en cuanto a la seguridad ya que durante la negociación un tercero podría intentar distintos ataques que conlleven a que las partes que negocian no lleguen a un acuerdo o éste podría obtener información acerca de la transacción así como del comportamiento en las partes (productos a comprar, precios, estrategia de negociación, etc.). Por estas razones, éste es uno de los aspectos que se tuvo en cuenta en el diseño del protocolo de pagos SPEED que presentamos en el Capítulo 3. Como última ventaja, cabe mencionar que el protocolo también permite establecer la cantidad a pagar, por lo que se satisface la propiedad de divisibilidad.

Una de las desventajas que se encuentra en el protocolo es que aparte del uso de claves públicas es necesario que el cliente obtenga previamente un ticket Kerberos para comunicarse con el vendedor. Este ticket Kerberos, que el cliente obtiene del vendedor, se obtiene de acuerdo a una versión modificada de Kerberos [280]. Por tanto, cada vez que el usuario quiera realizar compras con un vendedor, previamente tiene que obtener un ticket para comunicarse con tal vendedor. Esto supone que el vendedor tiene que adicionalmente implementar este tipo de sistema.

Otro aspecto negativo del protocolo es que el vendedor tiene que soportar una gran carga de trabajo debido a que todas las transacciones tienen que pasar por él, lo cual puede suponer que el vendedor se sature rápidamente. De esta forma, el sistema podría presentar problemas de escalabilidad. Éste es uno de los principales aspectos que buscaremos evitar con el protocolo de pagos SPEED.

eCash

eCash [269] es una de las pioneras en introducir, en 1990, un sistema de pago equivalente al dinero electrónico basado en monedas electrónicas y que se podía utilizar de forma completamente anónima. Para conseguir este anonimato, se utiliza una técnica conocida como *firma ciega* y que fue propuesta por Chaum [49, 50].

En este sistema participan tres entidades: cliente, vendedor y un banco. El banco tiene que estar en línea y es el responsable de generar las monedas junto con el cliente. Posteriormente, el vendedor contactará con el banco para comprobar si una moneda ha sido o no previamente utilizada. Debido a este modo de funcionamiento, se requiere que tanto el cliente como el vendedor tengan una cuenta con este banco.

En eCash, el proceso fundamental para garantizar el anonimato es el de retirada de dinero que lleva a cabo el cliente. En este paso el cliente crea las monedas con números de serie que son generados de forma aleatoria. Estas monedas se envían al banco para que éste las firme. Sin embargo, las monedas enviadas llevan ocultas el número de serie. A continuación, el banco carga el importe de las monedas en la cuenta del cliente y pasa a realizar una firma ciega. El banco utilizará una clave distinta para firmar las monedas de

cada valor. Una vez firmadas, el cliente obtiene las monedas con un número de serie visible.

Las monedas obtenidas pueden ser utilizadas para realizar pagos a los distintos vendedores. Cuando un vendedor recibe una moneda, debe comprobar que no ha sido previamente gastada tal y como hemos indicado antes. Posteriormente, cada cierto tiempo depositará todas las monedas recibidas.

La principal ventaja de este esquema es que garantiza el anonimato del cliente, ya que las monedas digitales se comportan como el dinero en efectivo, de forma que no contienen ninguna información relacionada con el cliente o con el producto que está adquiriendo. Este hecho se debe a que, como se mencionó anteriormente, el banco no sabe para quién genera las monedas. Además, el vendedor tampoco obtendrá información del cliente salvo que éste la revele durante el proceso de pago (por ejemplo, para la compra de productos nominales). En cambio, el anonimato del vendedor no está garantizado ya que éste tiene que depositar las monedas recibidas en una cuenta. Este protocolo ha sido probado mediante un proyecto piloto que duró tres años. Además, este protocolo ha servido como base a otras propuestas que han ido apareciendo posteriormente y que serán comentados en el siguiente apartado. De esta forma, el protocolo ha sido analizado en detalle y los niveles de anonimato y seguridad ofrecidos están contrastados [269].

A continuación mencionaremos los posibles problemas que presenta este esquema. En primer lugar, las monedas no son divisibles. Por tanto, para un determinado pago, un cliente tendría que poseer monedas por el importe exacto de la cantidad o poseer monedas de distintos tipos que le permitan sumar la cantidad solicitada, lo cual le podría implicar tener muchas monedas de distintos importes. En caso contrario tendría que contactar con el banco para obtenerlas. Este es el tipo de situaciones que trataremos de abordar con las propuestas que presentaremos posteriormente en el Capítulo 3.

Además, el sistema no proporciona la negociación segura ni un sistema de resolución de disputas entre los usuarios y el banco emisor de las monedas. Así podrían darse situaciones como las siguientes: que el banco indique que una moneda ya ha sido gastada cuando realmente no lo ha sido o el cliente podría utilizar para pagar más de una vez la misma moneda y negarlo. Finalmente, hay que mantener una lista con todas las monedas gastadas, la cuál puede ser muy larga e ir creciendo con el tiempo. Para evitar esta situación, habría que indicar una fecha de expiración de la moneda. Esto a su vez implica que las monedas del usuario que no han sido gastadas expiran. Para evitar la pérdida de las monedas no gastadas es necesario llevar a cabo procesos de actualización antes de que expiren, lo cuál también es un inconveniente.

Este protocolo además, para garantizar el no repudio, debería utilizarse con un protocolo que asegure el intercambio justo entre el producto y las monedas entregadas, ya que, en caso contrario, el vendedor podría obtener ventaja de esta situación, recibiendo las monedas en primer lugar y no proporcionando el producto posteriormente. En el Capítulo 3 veremos como el protocolo SPEED que proponemos puede evitar este tipo de situaciones.

Propuestas derivadas de eCash

Con el fin de mejorar las propiedades de eCash, han aparecido distintas propuestas derivadas de ésta. A continuación presentaremos una breve explicación de las más interesantes.

Fan et al. [99] en 2000 proponen un esquema basado en eCash y en funciones de resumen digital. Esta nueva propuesta introduce el uso de la fecha efectiva y la codifica como una serie de cadenas de resúmenes digitales (en concreto, seis, uno por cada dígito de la fecha: día, mes y año). De esta forma, el vendedor recibirá el interés entre la fecha efectiva y la fecha de depósito.

Chang y Lai [46], en 2003, proponen un esquema más flexible a la hora de adjuntar la fecha a la moneda. En este caso, se utilizan dos firmas ciegas, una para la moneda y otra para el resguardo de la fecha. El proceso de adjuntar la fecha se lleva a cabo por medio del uso de un sistema de correo electrónico seguro y no trazable.

Paycash [218, 219], que fue propuesto en 2003, también está basado en las ideas propuestas en eCash. Sin embargo, aporta varias novedades. La primera de ellas es que firma todos los registros de transacción y los integra en el sistema de pago con el objetivo de crear registros que no puedan ser manipulados. Otra diferencia es que el cliente, en vez de generar un número de serie, genera un par de claves que serán usados para la firma, aunque también incluye el número de serie. Ahora la moneda incluye una descripción (o un resumen de ella) de la transacción, el receptor de los fondos, un sello de tiempo y la firma de los datos de la transacción. Otra novedad importante es que se puede especificar la cantidad exacta por la que se quiere realizar el pago.

Popescu [226] (2005) propone un sistema de pago basado en monedas off-line y en el problema del logaritmo discreto en curvas elípticas. Esta propuesta es una mejora de otros esquemas previos, como los anteriormente mencionados, así como de otros descritos en [105, 224, 225]. La principal novedad de este esquema es que durante la fase de pago no es necesaria la participación del banco. Sin embargo, se requiere la participación de una tercero de confianza durante el proceso de retirada de monedas electrónicas. El esquema, al igual que sus predecesores, también garantiza el anonimato, aunque éste puede ser revocado en caso de doble gasto y así trazar tanto las monedas electrónicas como el propietario de la moneda electrónica. A pesar de estas mejoras, no se garantiza que, aunque el cliente lleve a cabo doble gasto, posteriormente las distintas entidades vayan a percibir las cantidades gastadas. Otro problema que presenta este esquema es que las monedas obtenidas no son divisibles y, por tanto, el cliente deberá de disponer de monedas por el importe a satisfacer, lo que en muchos casos implicará una conexión con el banco para obtener las cantidades deseadas.

Fan [100] propone en 2006 un esquema de moneda genérico similar a eCash, pero con la principal diferencia de que los propietarios del dinero electrónico pueden identificar las monedas perdidas. Para este propósito, este esquema sólo necesita incorporar dos operaciones de resumen digital. Por tanto, las ventajas y desventajas de este esquema con respecto a eCash son casi idénticas. Además, en esta propuesta se presenta un protocolo para garantizar el intercambio equitativo de bienes y de dinero electrónico.

D-Cash [148] (2007) propone un esquema de moneda electrónica basado en prepago, que tiene en cuenta las fechas de retirada del dinero, la fecha de la transacción y la fecha en la que se hace efectivo el pago. Estas fechas son interesantes para que las partes puedan cobrar el interés durante esos períodos. D-cash, a diferencia de las anteriores propuestas de Chang y Lai [46] o Fan et al. [99], también tiene en cuenta el interés de las monedas entre la fecha de retirada y la fecha efectiva. De esta forma, el vendedor puede pagar al cliente el interés de las monedas entre la fecha de retirada y la fecha efectiva. El banco, a su vez, pagará al vendedor el interés correspondiente al período entre la fecha de retirada y la fecha de depósito. Esta propuesta está basada en el uso de esquemas de firma ciega parciales, en pseudónimos y en firma DSA. Otra de las ventajas de este esquema es que puede ser fácilmente convertido para ser utilizado con criptografía de curva elíptica. En cuanto al resto de ventajas y desventajas comentar que serían similares a eCash, ya que la principal diferencia introducida es las fechas para el cobro de intereses de forma más precisa.

Payword

Este esquema de micropagos fue propuesto por Ronald Rivest y Adi Shamir [238] en 1997. Se trata de un sistema basado en crédito tanto para cliente, vendedor y broker. El broker se encarga de emitir certificados a los clientes, así como de pagar al vendedor cuando le presente las monedas recibidas de acuerdo a este esquema y que se denominan *paywords*. El vendedor provee los productos solicitados a cambio de *paywords* y, al final del día, presentará al broker los *paywords* recibidos junto con sus tokens de confirmación. El cliente se encarga de generar sus propias monedas y los tokens de confirmación para pagar a un vendedor a cambio de un producto.

Este sistema de micropagos está basado en cadenas de valores de resúmenes digitales, llamadas *paywords*. Cada *payword* representa una unidad de valor particular, es decir, una cadena de *paywords* es un conjunto de monedas del mismo valor, por ejemplo, podría representar 100 monedas de 1 céntimo de Euro. Los *paywords* son generados por el cliente y se puede crear por adelantado o en el momento de la compra. Para generar una cadena de *paywords*, el cliente elige un número aleatorio como el n-ésimo payword, w_n, que va a representar la semilla para generar el resto de los *paywords* de acuerdo con la siguiente regla:

$w_{i-1} = h(w_i)$ donde i=1,...,n

donde h es una función resumen digital como SHA-2 [201], que es resistente a colisiones e irreversible. El último valor calculado, w_0, no es parte de la cadena de *paywords* ($\{w_1, ..., w_n\}$), sino que constituye la raíz de la cadena e irá contenido dentro de un token de confirmación de la cadena específico para un cliente y vendedor. Este *token de confirmación de la cadena*, que firma el cliente, contiene la raíz (w_0), información del vendedor, el valor monetario de cada elemento de la cadena, la fecha de expiración y cierta información adicional. Por esta razón, se dice que el token de confirmación es específico para un cliente y un vendedor.

Los distintos procesos que tienen lugar en este esquema aparecen reflejados en la Figura 2.5. Antes de que cualquier transacción tenga lugar, el cliente debe obtener un certificado

emitido por el broker (paso 1). Este certificado sirve para autenticar la clave pública del cliente y además de esta clave contiene la identidad del broker, la fecha de expiración y otra información relacionada. El período de validez de este certificado normalmente es por un mes. Cuando el cliente encuentra un vendedor con el que quiere comerciar, genera una nueva cadena y un token de confirmación para dicha cadena.

Para efectuar la compra, el cliente envía al vendedor el token de confirmación de la cadena, un *payword* y la petición del producto (paso 2). Un *payword* se envía en claro, pero es autenticado por medio del token de confirmación de la cadena.

El vendedor verifica el token de confirmación y comprueba la fecha de expiración, así como que el token de confirmación ha sido realizado para él. También tiene que comprobar que el *payword* que le ha enviado el cliente todavía no ha sido gastado y que éste se corresponde con el importe del producto solicitado. Una vez comprobada esta información, almacena el *payword* como el último payword recibido (P_{last}) y envía el producto (paso 3).

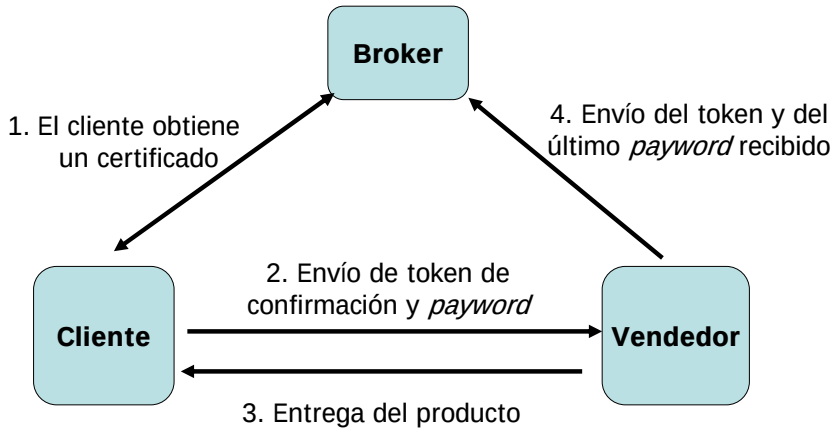


Figura 2.5. Esquema general de compra en Payword.

Si suponemos que el cliente cada vez compra un producto por el precio del valor un payword, el cliente irá enviado sucesivamente los distintos valores de la cadena hash (w_1, w_2, …). En cambio, si el artículo comprado cuesta más que el valor de un sólo *payword*, el cliente puede pagar enviado varios *paywords* a la vez. Supuesto que el último *payword* sin gastar es w_3, si cada *payword* tiene un valor de 1 céntimo de Euro, y el artículo cuesta 5 céntimos, el cliente tendría que enviar los paywords desde w_3 a w_8. Sin embargo, no es necesario enviarlos todos, con enviar w_8 es suficiente. Para comprobar el valor recibido, el vendedor aplica la función hash a w_8 hasta llegar al último *payword* utilizado por el cliente anteriormente (en este caso w_2). En este caso, realizará 5 operaciones de resumen digital. Por tanto, habrá recibido 5 monedas.

El vendedor para cobrar las monedas recibidas como pago, al final del día, sólo necesita enviar el token de confirmación de la cadena y el último *payword* recibido (P_{last}). El broker verifica el token de confirmación y se asegura que se puede obtener w_0 después de

aplicar el resumen digital *last* veces al *payword* P_{last}. Si todo es correcto, el broker descuenta la cantidad de la cuenta del cliente para ingresársela al vendedor.

La principal ventaja de Payword es la sencillez de sus mensajes y que sólo utiliza una operación criptográfica de clave pública por cadena de *paywords*. Otra de las ventajas que presenta es que para las compras la verificación es descentralizada, es decir, no tiene que estar presente el broker, sólo al final del día el vendedor establecerá contacto con el broker. Por tanto, las transacciones de pago se pueden llevar a cabo más rápidamente que si esta entidad tuviera que participar ya que, en tal caso, podría tener que soportar un elevado número de transacciones por segundo además de que podría suponer un punto de fallo en el sistema. Este tipo de consideraciones también se tuvieron en cuenta nuestras propuestas de protocolos de pago que presentamos en el Capítulo 3, principalmente, en PURSE-COIN.

Como última ventaja mencionar que el cliente no tiene que pagar por adelantado, sólo pagará al final del día la cantidad que haya gastado. Debido a estas propiedades podemos decir que el esquema es eficiente y escalable.

Las desventajas que presenta son que no puede garantizar el no repudio [3] ya que no se indica en el protocolo qué es lo que desea el cliente. Esto nos lleva a que la resolución de disputas de gasto entre el cliente y el vendedor es teóricamente imposible, ya sea porque el comprador es deshonesto o porque el deshonesto es el vendedor. También presenta el mismo problema que esquemas anteriores y es que no se pueden prevenir los fraudes que cometa el vendedor de la no entrega de la información o de que esta información sea incorrecta, ya que tampoco se especifica lo que se entrega, ni se adjunta prueba de ello. Por tanto, no se garantiza el no repudio e intercambio justo.

Otro problema que presenta es que podemos estar desperdiciando nuestros recursos en la generación inútil de monedas si lo que hacemos es tratar con muchos vendedores y con cada uno de ellos gastamos muy poco. En este esquema, el principal problema que se plantea es que el broker asume la responsabilidad del pago, ya que él es el que responde por los clientes a los que firma, de forma que debe tener extremas precauciones a la hora de firmar a los clientes, ya que como el dinero lo genera el cliente en un día, en el caso de que dos o tres clientes actúen de forma fraudulenta, puede llegar a tener pérdidas importantes debido a que la generación del dinero es sencilla y los clientes podrían exceder su crédito [3].

Esquemas derivados de Payword

En esta sección comentamos varios esquemas basados en las ideas propuestas por Payword, es decir, basadas en el uso de cadenas de resúmenes digitales.

La primera de estas propuestas es NetPay [65, 66, 67] y fue introducida en 1999. Netpay está basado en débito. En este sistema participan cliente, vendedor, broker NetPay y un banco. Este sistema está basado en Payword aunque difiere de éste en que utilizan lo que llaman una *piedra de toque* o *token* que está firmada por el broker y que representa un índice de moneda electrónica firmado por los vendedores, que es pasada de vendedor a vendedor de forma que la misma cadena de paywords puede ser utilizada por

varios vendedores. Además, se utiliza por el vendedor para impedir el doble gasto de los clientes y para resolver disputas entre vendedores.

Entre las ventajas de este esquema podemos mencionar que al estar basado en débito, el vendedor está seguro de que cobrará las monedas recibidas y el broker tiene un mayor control del dinero generado. Además, y también a diferencia de Payword, la cadena de resúmenes digitales puede ser utilizada con más de un vendedor. En el caso de que las monedas fueran interceptadas cuando son enviadas a un vendedor, éstas sólo podrían ser utilizadas con ese vendedor en particular.

Sin embargo, esta propuesta sigue adoleciendo del mismo problema que presentaba el esquema de Payword, y es que, como todas las monedas son del mismo importe, no resuelve bien el problema de la divisibilidad. Además, cada vez que se quiere pagar a un nuevo vendedor con la misma cadena utilizada con otro vendedor en el último pago, se requiere la participación de este último vendedor. Es decir, en el proceso de pago ocasionalmente se requiere la participación de un tercero.

Zangkoi et al. [326] en 2004 introducen una propuesta que, a diferencia de Payword, está basada en prepago, con la posibilidad de que cada uno de los valores de la cadena represente distintos importes. Además, permite que una misma cadena pueda ser utilizada con distintos vendedores. Así, se requiere que el cliente, antes de realizar el pago, contacte con el broker o entidad financiera para obtener las monedas que se utilizarán en ese pago en concreto. El broker entonces confirmará la cadena de resúmenes digitales a utilizar y generará una clave de sesión de un solo uso que compartirán cliente y vendedor; además, enviará una autorización al vendedor. La ventaja de este esquema frente a *Payword* es que, al estar basado en prepago, supone menos riesgo para vendedor y broker. Sin embargo, requiere la participación del broker en los procesos de pago, ya que por cada orden de pago tiene que emitir una autorización de pago.

En 2006 Hwang y Sung [139] revisan varios de estos protocolos (incluido Payword), tales como el protocolo propuesto por Hwang y Lin [138], PayTree [151] y la propuesta de Kim y Lee [156]. Además, proponen un nuevo protocolo como mejora a las soluciones anteriormente propuestas. Su propuesta está basada en débito, puede ser utilizada con distintos vendedores y garantiza el anonimato.

En este protocolo, como paso previo al pago, es necesario un proceso de registro donde se crea una clave compartida entre el cliente y el vendedor. Además, la generación de la cadena requiere el contacto con el broker para un proceso de firma ciega. La participación del broker también es necesaria durante el proceso de pago para conocer la clave a utilizar con el vendedor y crear una clave de sesión para las tres entidades.

Esta propuesta tiene varias ventajas sobre Payword. En primer lugar, el esquema está basado en débito; por lo tanto, el riesgo es menor. Además, la cadena puede ser utilizada con varios vendedores. Así como requiere almacenar menos información referente al pago mientras no se recibe éste. Finalmente, el número de operaciones criptográficas de clave pública a realizar es menor. Como principal desventaja podríamos señalar que el broker tiene una mayor participación en el proceso de pago.

Comparativa de sistemas de pago electrónico B2C

Una vez presentados los principales protocolos de pagos, a continuación, mostraremos una tabla (ver Tabla 2.1) que resume las características de cada uno de ellos. Esta tabla se ha construido en base a las distintas características de los protocolos de pago que mencionamos anteriormente. Además, se han indicado otras propiedades que consideramos que pueden ser de utilidad a la hora de comparar los protocolos. Estas propiedades adicionales son: si está basado en cuenta o en moneda, el tipo de modelo de pago soportado (crédito, débito o prepago), el tipo de criptografía utilizada (asimétrica y/o simétrica), si contempla la negociación y/o distribución segura del producto o servicio electrónico a adquirir y, finalmente, si existe alguna implementación que sirva para probar la utilidad del protocolo a los usuarios.

Características / Sistema de pago	Basado en Cuenta (C) / Moneda (M)	Crédito (C) / Débito (D) / Prepago (P)	Criptografía Asimétrica o Curvas elípticas (ECC)	Criptografía Simétrica	Tipo de validación: on-line (On), off-line (Off) o semi-online (SO)	No Repudio	Intercambio equitativo	Atomicidad	Negociación Segura	Distribución segura	Divisibilidad	Anonimato	Implementación
SSL/TLS	C	x_1	x	x	On				x_2	x_2	x		x
SET	C	C	x	x	On	x_3					x		x
3D	C	C	x	x	On						x		x
NetBill	C	D	x	x	On	x_4	x	x	x	x	x		x
Ecash	M	P	x		On							x	x
Fan et al.	M	P	x		On							x	
Chang y Lai	M	P	x		On							x	
D-Cash	M	P	x		On							x	
PayCash	M	P	x		On							x	
Fan et al.	M	P	x	x	On		x			x		x	
Popescu	M	P	x		Off							x	
Payword	M	C	x	x	Off								
Hwang y Sung	M	D	x	x	Off								
NetPay	M	D	x	x	SO								x

Tabla 2.1. Comparativa de los principales protocolos de pago.

Notas:

x_1: Normalmente, SSL/TLS se utiliza para pagos con tarjeta. El modelo de pago dependerá del tipo de tarjeta utilizada. Normalmente, crédito o débito.

x_2: De la misma forma que SSL/TLS se utiliza para enviar la información de pago, también se podría utilizar para realizar la negociación y/o distribución de productos electrónicos.

x_3: Garantiza el no repudio. Aunque se han encontrado diversos ataques que llevan a que no se satisfaga en determinadas circunstancias.

x_4: Se ha encontrado un ataque a la atomicidad.

Después del análisis de los distintos sistemas de pagos podemos extraer varias conclusiones. Tal y como se refleja en la Tabla 2.1 cabe destacar que sólo hay un esquema que tenga en cuenta la seguridad en las principales fases del proceso de compra (negociación, pago y distribución), en este caso, NetBill. Por este motivo, en nuestra propuesta de protocolo de pago SPEED intentamos seguir el modelo propuesto por éste.

Además, en general, en las propuestas presentadas no se garantiza el no repudio, el intercambio equitativo y la atomicidad. Así, en la mayoría de los casos el vendedor está en ventaja sobre los clientes, lo que provoca desconfianza en éstos tal y como se menciona en [118].

De los esquemas presentados, sólo Netbill ofrece una solución global al problema. Sin embargo, esta solución supone una importante participación del vendedor en el proceso de pago que puede limitar la escalabilidad de la solución ofrecida. Además, también se han encontrado un ataque a la propiedad de atomicidad. Por tanto, nuestra propuesta de pago SPEED buscará solucionar este problema a la misma vez que ofrecer características similares a éste.

Por otro lado, existen soluciones, principalmente basadas en moneda electrónica, que aunque no garantizan el intercambio equitativo, podrían solucionar este problema mediante su combinación con un protocolo de este tipo. Así la moneda se intercambiaría por el producto elegido. Sin embargo, en otros protocolos, como en SET, esta solución no sería factible, ya que el pago supone el intercambio de varios mensajes de pago.

La mayoría de las propuestas están basadas en el uso de una tercera parte que, aunque en muchos casos supone un incremento en los costes y en los tiempos de ejecución del protocolo, sin embargo, aumenta la seguridad del sistema. Finalmente, la divisibilidad es un aspecto que no se ha considerado fundamental en muchas de las propuestas analizadas.

Por tanto, de estas conclusiones se puede desprender que sería deseable un protocolo de pago que contemplase la seguridad en todas las fases de la compra, garantizando el no repudio, el intercambio equitativo y la atomicidad con una menor carga para el vendedor.

2.2.3. Las tarjetas inteligentes como impulsoras de los medios de pago

A día de hoy, las tarjetas de crédito o débito basadas en banda magnética son uno de los sistemas de pago más utilizados. Sin embargo, la seguridad de estas tarjetas es reducida y esto ha provocado que el nivel de fraude cada vez sea mayor [6, 187]. En una tarjeta de banda magnética, cualquiera que posea el equipo necesario puede leer, borrar o reescribir la información almacenada en la banda [234]. Este hecho hace posible que se pueda clonar este tipo de tarjeta.

Las tarjetas inteligentes nacen como una evolución de las tarjetas de banda magnética para superar los problemas que éstas presentan y así ofrecer un sistema más seguro de almacenamiento. Las principales áreas que se han visto beneficiadas de esta tecnología son:

- La telefonía, que ha incorporado el uso de esta tarjeta en los teléfonos móviles para identificar al usuario, así como en tarjetas de prepago para realizar llamadas en cabinas telefónicas.
- Sistemas de pago electrónico, tanto para proteger información como nuestro número de tarjeta de crédito y autorizar pagos, como para implementar sistemas de pago basados en monedero electrónico.
- Sistemas de control de acceso, para establecer sistemas de autorización más seguros.
- Sistemas sanitarios, para identificar a los usuarios y almacenar expedientes médicos.
- Firma electrónicas y sistemas de identificación electrónica, de forma que las tarjetas inteligentes se han incorporado como parte del documento nacional de identidad (DNI) o pasaporte electrónico de algunos países como España, Finlandia, etc.
- Otras áreas o ámbitos que tengan características comunes con los que acabamos de describir.

Técnicamente una tarjeta inteligente consiste en una tarjeta de plástico PVC que contiene un circuito integrado y que tiene componentes para transmitir, almacenar y procesar datos. Los datos pueden ser transmitidos usando los contactos de la superficie de la tarjeta o mediante campos electromagnéticos sin ningún tipo de contactos. Las características fundamentales y las funciones de las tarjetas están especificadas en la familia de estándares ISO 7816 [142]. También se pueden encontrar más detalles en [234]. De entre las distintas características, la más reseñable es que las tarjetas inteligentes se consideran dispositivos a prueba de manipulaciones y ataques (tamper resistance) [157, 234, 269]. Es decir, que la tarjeta así como sus datos y el comportamiento algorítmico no puede ser manipulado, copiado o revelado a, o por medio de, una persona no autorizada.

Estas tarjetas ofrecen varias ventajas sobre las de banda magnética:

- Mayor capacidad de almacenamiento.
- Mayor protección de los datos contra accesos no autorizados.
- Mayor nivel de fiabilidad.
- Vida útil más larga.

- Pueden soportar más de una aplicación (multi-aplicación).
- Mejor respuesta a cambios en las aplicaciones.
- Se podrían reducir costes debido a que muchos procesos de autorización, en vez de realizarse en línea, se podría realizar de forma fuera de línea (localmente) debido a la seguridad aportada por el chip.

El principal problema que presenta el uso de las tarjetas inteligentes es los importantes costes de migración de tarjetas de banda magnética a estas nuevas tarjetas. En primer lugar, el precio del circuito integrado es mucho más caro que la banda magnética, coste que tienen que soportar los emisores. En segundo lugar, se necesitan nuevos tipos de terminal para poder utilizar esta clase de tarjetas. Estos son los principales motivos por lo que, a pesar de las ventajas ofrecidas por este tipo de tarjetas, el proceso de migración esté siendo muy lento [231].

Inicialmente, el desarrollo de aplicaciones, como las descritas anteriormente, implicaba la creación de software específico para el hardware concreto de la tarjeta. Además, una vez desarrollado el software no había posibilidad de actualizarlo en la tarjeta, si no que las actualizaciones suponían el reemplazo de la tarjeta por otra con el nuevo software.

En respuesta a esta problemática, se decidió crear una tarjeta que pudieran ejecutar programas que fueran independientes de la plataforma basándose en el uso de un intérprete. La solución fue llamada Java Card [52, 124, 234, 283] ya que iba a permitir el desarrollo programas para tarjetas inteligentes en un lenguaje de alto nivel como Java.

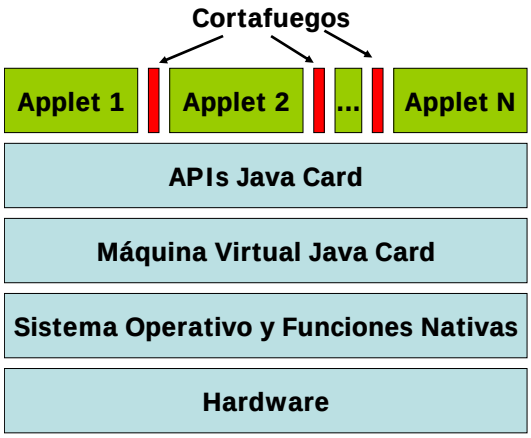


Figura 2.6. Arquitectura Java Card.

En la Figura 2.6 se pueden observar los distintos elementos que forman parte de la arquitectura de una tarjeta Java Card. En la base de la arquitectura está el hardware. Sobre éste se construye una capa que contiene el sistema operativo y las funciones nativas. Esta capa es la base sobre la que se establece la máquina virtual Java Card. Esta máquina virtual permite alojar al entorno de ejecución de la Java Card. Este entorno de ejecución ofrece distintas APIs de programación sobre las que se construirán las diferentes aplicaciones que, en la tarjeta Java Card, se denominan *applets*. Los applets están

protegidos entre sí por un mecanismo conocido como cortafuegos que impide que un applet pueda acceder a los datos o información de otro applet a menos que éste lo haya autorizado. Existe una aplicación especial denominada gestor de la tarjeta (card manager) que es la responsable de la instalación y gestión de los distintos applets que co-existirán en la tarjeta.

Las tarjetas inteligentes a día de hoy constituyen uno de los elementos fundamentales necesarios para garantizar el no repudio de las transacciones y el comercio electrónico. De hecho, la directiva 1999/93/EC del parlamento europeo establece las bases que permiten equiparar la firma electrónica a la firma manuscrita, un análisis más en profundidad se puede encontrar en [243, 244, 257]. En dicha directiva, para que la firma electrónica se considere equivalente a la firma manuscrita se tienen que satisfacer las siguientes condiciones:

- Estar ligada de forma única al firmante.
- Ser capaz de identificar al firmante.
- Ser creada usando medios que el firmante puede mantener bajo su exclusivo control.
- Estar ligada a los datos a los cuales está asociada de tal manera que cualquier cambio de los datos es detectable.
- Estar generada usando un dispositivo seguro de creación de firma (SSCD – Secure Signature Creation Device). Los requisitos que deben satisfacer estos dispositivos están descritos en [95]. Algunos de ellos son tarjetas criptográficas que cumplen determinados niveles de seguridad que están determinados en la especificación que acabamos de mencionar y que se conoce como EAL 4+. Un ejemplo sería el actual DNI electrónico (DNI-e).
- El certificado está emitido por una autoridad de certificación (o prestador de servicios de certificación) reconocido. Una autoridad de certificación se considera reconocida si satisface determinados requisitos de seguridad y ofrece determinadas garantías en cuanto al ciclo de vida de los certificados que emite.

En resumen, para obtener una firma electrónica legalmente equivalente a la manuscrita tendríamos que utilizar un dispositivo como una tarjeta inteligente que sea conforme a las especificaciones EAL 4+ y que el certificado utilizado para realizar la firma estuviese emitido por una autoridad de certificación reconocida.

Esta directiva establece el marco legal necesario para dar cobertura a los procesos de negocio en la red. Este hecho provocará que en un futuro el uso de las tarjetas inteligentes se extienda de forma paulatina. Es más, este marco puede dar lugar a una mayor extensión del comercio móvil, ya que todos los teléfonos móviles incluyen una tarjeta inteligente denominada SIM (Subscriber Identifier Module) y que podría satisfacer los requisitos anteriormente mencionados. Un análisis más detallado de las distintas soluciones de firma móvil que podrían ser utilizadas de acuerdo con este marco aparece en [257, 258].

2.2.4. Estándares de pago basados en tarjeta inteligente

En esta sección describimos los estándares más importantes de pago basados en tarjeta inteligente, es decir, WG10 [44], CEPS [45] y EMV [89, 90, 91, 92, 231]. Existen otras propuestas de pago basadas en tarjeta inteligente tales como [48, 126]. Sin embargo, no analizaremos aquí dichas propuestas porque en ellas la tarjeta inteligente simplemente se utiliza como un contenedor de claves privadas y/o monedas que facilita la movilidad e incrementa la protección de esta información. En cualquiera de las soluciones planteadas en la Sección 2.2.2 también se podría incorporar la tarjeta para tal este propósito. Sin embargo, el objetivo de esta sección es analizar las soluciones en las que la tarjeta se utiliza como monedero de forma que pueda generar tokens de dinero electrónico que se van a poder utilizar en distintos entornos. Normalmente, se entiende por *monedero electrónico* a aquellas tarjetas inteligentes que llevan una aplicación de pago basada en contador y en prepago. Nosotros extendemos esta definición, entendiendo por monedero electrónico aquellas soluciones de tarjeta inteligente basadas en contador que pueden emitir dinero electrónico por un valor distinto al almacenado.

Puesto que las tarjetas inteligentes son más seguras que las tarjetas de banda magnética, el objetivo es analizar estos estándares de cara a ver su posible uso para realizar pagos en la Web. Por tanto analizaremos la seguridad de las transacciones de pago que permiten teniendo aspectos tales como el no repudio, intercambio equitativo, etc. También se analizarán otras propiedades como el anonimato o la participación de una tercera parte durante el proceso de pago. Este análisis nos permitirá comprender los distintos problemas que estas soluciones presentan de cara a su uso en la Web y que nos ha llevado a la propuesta de las soluciones de pago basadas en monedero que presentamos en el Capítulo 3.

CEN/TC224/WG10 prEN1546

El estándar EN 1546 [44], más conocido como monedero WG10, fue diseñado por el Comité Europeo de Normalización (CEN) dentro de su comité técnico 224 (TC244). Este estándar nace con el objetivo de proporcionar un sistema de monedero electrónico multisector (Intersector Electronic Purse o IEP).

El modelo de negocio del monedero WG10 se basa en prepago, de forma que para realizar pagos es necesario que previamente éste haya sido cargado con dinero electrónico. Este proceso de carga se puede realizar o bien en las entidades financieras que los emiten, o bien en puntos de carga diseñados para tal efecto (por ejemplo, cajeros automáticos o máquinas top-up o de recarga). En el proceso de carga, el usuario deposita dinero en efectivo o contra su cuenta de bancaria/crédito y, por el mismo importe del depósito se incrementa el saldo del monedero.

WG10 está basado en el uso de un terminal con un módulo SAM (Secure Authentication Module – Módulo de Autenticación Segura) que posee una clave maestra como elemento central del sistema. Cada monedero poseerá una clave diversificada que se obtendrá a partir del número de serie de la tarjeta y la clave maestra del módulo SAM. Cualquier operación de incremento o decremento del saldo del monedero requerirá un

intercambio de mensajes entre el módulo SAM y la tarjeta con la que realizar la operación. Los mensajes intercambiados están “firmados” por medio de clave simétricas utilizando el algoritmo DES o 3DES. Sin embargo, para dotar de mayor seguridad al sistema, en ningún momento se firma la información con la clave diversificada de la tarjeta, sino que se genera una clave de sesión que utilizarán tanto el módulo SAM como el monedero (esta clave de sesión se calcula a partir del número de transacción y de la clave diversificada contenida en el monedero).

En el proceso de incremento del saldo, el monedero se autentica frente al módulo SAM (mensaje denominado S1). A continuación, el módulo SAM se autentica frente al monedero y autoriza la transacción (mensaje que se denominada S2). Finalmente, el monedero confirma que la transacción se ha realizado satisfactoriamente. El proceso de incremento de saldo es similar y, con respecto a la anterior transacción, se mantienen el mismo tipo de mensajes (S1, S2 y S3). La única diferencia es la información intercambiada en los mensajes.

La seguridad de este esquema de monedero está basada en el uso de estos módulos SAM. Éstos son los que autorizan cualquier operación mediante la firma de los comandos con unas claves simétricas que sólo conocen el módulo SAM y el monedero concreto en el que se lleva a cabo la transacción. Este sistema de monedero, al estar diseñado sobre todo para su utilización en máquinas de vending, no garantiza el no repudio de la transacción ni el intercambio equitativo.

CEPS

La especificación de monedero electrónico común (Common Electronic Purse Specification - CEPS) [45] fue publicada en 1999. En esta especificación se definen los requisitos para un sistema de monedero electrónico interoperable internacionalmente. Además, en CEPS se requiere compatibilidad con las especificaciones EMV para tarjetas inteligentes. Este monedero está basado en CEN/TC224/WG10 [44] que ya fue comentado anteriormente. En esta especificación, además de definir las operaciones de carga y decremento, se definen otras operaciones como son el decremento incremental, revocación de un decremento, cancelación de la última compra y cambio de divisa.

El monedero CEPS, a diferencia del monedero WG10, utiliza criptografía asimétrica basada en RSA para el proceso de compra e introduce el uso de los certificados para los distintos elementos del sistema. En este esquema la autenticación se basa en el intercambio de certificados para obtener información sobre las claves con las que autenticar a la otra parte. La criptografía simétrica basada en 3DES se utiliza para el proceso de carga que se realiza de forma on-line.

En el proceso de carga, el dispositivo de carga establece una conexión con el emisor de la tarjeta, actuando como proxy en la transacción. Las tarjetas CEPS soportan dos tipos de carga. En el primer tipo, la carga se realiza a partir de una cuenta que el usuario tiene en la entidad que emitió la tarjeta. En el segundo tipo, la carga se puede realizar por otras fuentes que no sean el emisor de la tarjeta.

Las fases que forman parte del proceso de carga son las siguientes:

- Intercambio de divisas. El dispositivo de carga y la tarjeta intercambian las divisas que ambos soportan.
- Autenticación del usuario por medio de su PIN. El usuario se autentica frente a su tarjeta.
- Autenticación de la tarjeta frente al emisor (comando S1).
- Autenticación del emisor frente a la tarjeta (comando S2).
- Confirmación del incremento del saldo de la tarjeta (comando S3).

La seguridad del proceso de CEPS ha sido ampliamente estudiada y analizada con herramientas de modelado y verificación automática como en [149, 150, 155]. Jürgens y Wimmel en [150] presentan un potencial problema de seguridad cuando un atacante intenta autenticarse con el módulo SAM de compra. Este ataque, sin embargo, no tendría lugar por medio de protocolos de Internet. Por tanto, es un problema más relacionado con la seguridad física.

Kim et al. [155] presentan un posible problema de contabilidad del dinero (el dinero no es creado ni destruido en el proceso de una transacción electrónica entre el cliente y el vendedor) cuando hay fallos en la red de comunicaciones. Por ejemplo, un cliente carga dinero electrónico en la tarjeta desde el banco e intenta utilizar ese dinero para pagar a un vendedor. Si la red fallase en el tránsito del dinero electrónico desde el cliente al vendedor, el primero no estará convencido de si fue gastado correctamente. Este problema, como estos autores proponen, se podría solucionar con un registro de transacciones en la entidad bancaria. Otros aspectos a tener en cuenta es que este esquema de monedero no garantiza el no repudio ni el intercambio equitativo.

EMV

EMV [89, 90, 91, 92, 231], cuya última especificación apareció en 2007, es un estándar de facto en el área de sistemas de pago basados en crédito/débito propuesto por Europay, Mastercard y VISA. La aparición de EMV surgió de la necesidad de sustituir las tarjetas de crédito actuales (basadas en banda magnética) por otras tarjetas que proporcionen más seguridad, en este caso las tarjetas inteligentes. Este hecho está motivado porque las tarjetas de crédito son fácilmente manipulables y están ocasionando importantes niveles de fraude (por ejemplo, en UK, en 2007, se estima que las cifras alcanzadas están entorno a 428 millones de libras esterlinas [6]).

En EMV se definieron un conjunto común de estándares para aplicaciones de pago basadas en tarjeta inteligente. Uno de los principales objetivos de este diseño era garantizar la interoperabilidad y permitir el uso de múltiples aplicaciones de pago y de diferentes operadores de pago en la misma tarjeta inteligente. De esta forma, terminales de las distintas organizaciones serían capaces de procesar pagos de otras organizaciones sin tener que conocer previamente la estructura de la tarjeta de tal organización. Así, por ejemplo, se facilitaba que el terminal para las tarjetas VISA y Mastercard fuera el mismo.

Otra de las características que ofrecen las tarjetas EMV es la posibilidad de realizar pagos remotos en entornos de comercio electrónico o comercio móvil. En este escenario de pago remoto, se supone que la transacción se realiza por medio del protocolo de pagos SET.

Las distintas fases por las que pasaría una transacción en EMV son las que indicamos a continuación:

- Selección de la aplicación con la que se realizará la transacción.
- Inicialización de la transacción, donde se intercambian los datos que formarán parte de dicha transacción.
- Autenticación de la tarjeta.
- Autenticación del usuario. Una vez autenticada la tarjeta, se procede a autenticar o verificar al usuario.
- Funciones de gestión de riesgo. Determinan si se ha de realizar una autorización on-line de la transacción o no.
- Autenticación on-line. Este proceso se realiza si en el anterior paso se determinó que debía de ejecutarse.
- Finalización de la transacción.

Para poder realizar una operación de crédito/débito es necesario poseer la tarjeta inteligente y, además, conocer el número PIN que permite su acceso.

EMV define varias clases de protocolos de seguridad: Autenticación de la tarjeta de forma off-line (Off-line CAM – Card Authentication Method), cifrado del PIN para soportar la verificación del propietario de la tarjeta (CVM – Cardholder Verification Method), generación de criptogramas para proteger las transacciones y mensajes seguros en las comunicaciones entre la tarjeta y su emisor.

Para la autenticación de la tarjeta definen tres métodos: estática (SDA – Static Data Authentication), dinámica (DDA – Dynamic Data Authentication) y combinada (CDA – Combined Data Authentication).

En la autenticación estática (CDA) el emisor de la tarjeta almacena determinada información cuando personaliza la tarjeta. Esta información será enviada durante el proceso de autenticación. Sin embargo, esta autenticación es débil porque la copia o los reenvíos son posibles, ya que la información que se envía siempre es la misma. Esta solución se optó para aquellos casos en los que las entidades bancarias querían proporcionar tarjetas inteligentes pero no querían aún introducir tarjetas inteligentes criptográficas que suponen un mayor coste.

La autenticación dinámica (DDA) permite la autenticación de la tarjeta en función de datos dinámicos (un reto) que generará el terminal. Esta autenticación básicamente es la firma de los datos que solicita el terminal.

Finalmente, la autenticación combinada (CDA) es una variación de la autenticación DDA, en la que la tarjeta, aparte de firmar el reto que le solicita el terminal, también firma la información relativa a la transacción.

EMV ofrece dos métodos para autenticar al usuario frente a la tarjeta, que es lo que se denominada método de verificación del poseedor de la tarjeta (CVM – Cardholder Verification Method). El primer método, denominado off-line, consiste en la verificación del PIN en el punto de venta. En el segundo método, denominado on-line, la verificación del PIN tiene lugar en el emisor de la tarjeta.

En EMV también se definen funciones para la gestión del riesgo que pueden ejecutarse tanto en la tarjeta como en el terminal. Básicamente, la gestión del riesgo implica realizar una autorización on-line de la transacción, de forma que, si alguna de las dos partes lo solicita, ésta debe llevarse a cabo. Los criterios que podría seguir el terminal para solicitar una autorización on-line son: elección aleatoria, cada X número de transacciones y/o a partir de una cierta cantidad. En la tarjeta los criterios podrían ser: cada X transacciones o después de acumular cierta cantidad de dinero en compras off-line. Por tanto, es necesario poder establecer en cualquier momento una conexión on-line con la entidad bancaria para la realización de una operación de pago.

Como se desprende de los distintos modos de funcionamiento soportados, EMV ofrece distintos niveles de seguridad de forma que se pueda ir realizando un proceso de migración paulatino. Suponiendo el uso de CDA/DDA, podemos decir que esta propuesta garantiza el no repudio del cliente, sin embargo, no garantiza el no repudio del vendedor o del terminal ya que no se generan las evidencias necesarias [174]. Además, en la autenticación del emisor, no se incluye información de la identidad del emisor. En general, las transacciones no requieren la conexión con una tercera entidad, salvo que el mecanismo de gestión del riesgo de la tarjeta o del terminal indique que una determinada transacción tenga que ser efectuada on-line. En las transacciones, el cliente puede especificar la cantidad exacta por la que se desea pagar, aunque estas cantidades son adecuadas para (macro)pagos. Este protocolo tampoco garantiza el intercambio equitativo, de forma que el cliente podría realizar el pago y, posteriormente, no recibir el producto solicitado. Finalmente, mencionar que el proceso de pago no es anónimo, aunque la entidad financiera no conocerá los datos de la compra.

Comparativa de los estándares de pago basados en tarjeta

Una vez realizado un análisis y descripción de las principales propuestas de pago basadas en tarjeta inteligente, pasaremos a mostrar una tabla (ver Tabla 2.2) que muestra a modo de resumen las principales características de cada una de las propuestas.

La tabla, por cada sistema de pago, indica el modo de pago seguido (crédito, débito, prepago), el tipo de criptografía utilizada (asimétrica y/o simétrica), si es necesario el uso de un módulo de seguridad en el vendedor para el proceso de pago, si garantiza el no repudio y el intercambio equitativo, si es divisible el esquema propuesto, si garantiza el anonimato y, finalmente, si es necesaria o no la participación de una tercera parte de confianza, como podría ser un banco o un broker.

La principal conclusión que podemos extraer de esta tabla y del análisis realizado de cada una de las propuestas es que ninguna de ellas garantiza el intercambio equitativo y que sólo una de ellas garantiza el no repudio (parcialmente). El principal motivo de estas carencias es que la mayoría de las propuestas están pensadas principalmente para su uso

en máquinas de vending o en puntos de venta, donde la comunicación entre la tarjeta y el vendedor es realizada en local, ya que se realiza directamente sobre la máquina. Sólo EMV contempla la posibilidad de su uso en Internet aunque sigue siendo necesaria la integración de un dispositivo con un SAM para el pago. Otro de los aspectos que no ha sido considerado en la mayoría de ellas es el anonimato, ya que en este tipo de transacciones, al ser realizadas por medio de certificados identificativos, el banco obtiene la información acerca de la identidad del cliente involucrado en la transacción, salvo en WG10. Por otro lado, en general, la entidad financiera no obtiene la descripción de los productos o servicios proporcionados por el vendedor.

El hecho de que el uso de las tarjetas inteligentes con una aplicación de pago aunque es más seguro que una tarjeta de crédito no esté extendido para llevar a cabo pagos en Internet puede deberse a dos motivos. En primer lugar, al hecho de que hasta ahora las tarjetas no eran muy conocidas y por tanto no estaban muy extendidas. Ahora con la implantación progresiva del DNI electrónico y las tarjetas bancarias con EMV este problema podría empezar a solucionarse. Segundo, el hecho de que para realizar un pago con tarjeta en Internet requiera que los vendedores integren en sus soluciones de pago la comunicación entre la tarjeta inteligente y el módulo SAM correspondiente. Esta integración es compleja y además puede llegar a hacer que el pago sea lento.

Por tanto, para solucionar este problema debería proporcionarse una solución de pago basado en monedero que facilitara su uso en Internet evitando el uso de conexiones on-line con un tercero y el uso de módulos SAM en el vendedor. Esta solución a su vez debería permitir también su uso en máquinas de vending o en puntos de venta para seguir garantizando las mismas ventajas que los monederos actuales.

Características / Sistema de pago	Modelo de Pago	Criptografía Asimétrica o Curvas elípticas (ECC)	Criptografía Simétrica	Módulo SAM	No repudio	Intercambio equitativo	Divisibilidad	Anonimato	Participación TTP
WG10	P		x	x			x	x	
EMV	D	x_1	x		x_2		x		x_3
CEPS	P	x	x	x			x		

Tabla 2.2. Comparativa de los sistemas de pago basados en tarjeta inteligente.

Notas:

x_1: Depende de si el esquema es SDA/CDA/DDA. CDA/DDA requieren criptografía asimétrica.

x_2: Garantiza el no repudio del cliente, pero no el del vendedor.

x_3: Depende del mecanismo de gestión del riesgo.

2.2.5. Conclusiones

En esta sección (2.2) de sistemas de pagos hemos analizado las principales propuestas de pago, tanto basadas en tarjeta inteligente como aquellas que no están basadas en este tipo de dispositivos. En estas últimas, el principal problema surge a la hora de gestionar el dinero electrónico y cómo utilizarlo fácilmente en distintos entornos. En este sentido, las tarjetas inteligentes ofrecen un sistema de almacenamiento seguro que puede convertirse en el equivalente al monedero que tradicionalmente llevamos en nuestro bolsillo. Pero las tarjetas no sólo se pueden utilizar como un simple contenedor seguro y, por eso, han aparecido propuestas de pago basadas en tarjeta inteligente. A día de hoy éstas principalmente se utilizan en entornos de máquinas de vending. Su uso para pagos en Internet es limitado debido a que se requiere que los vendedores integren dispositivos con módulos SAM en sus aplicaciones de comercio para poder realizar el proceso de pago con estas tarjetas. Este proceso puede ser lento debido a los mensajes que hay que intercambiar entre estos dos dispositivos. Otro de los factores que limitaba su uso era la escasa extensión del uso de las tarjetas inteligentes. Sin embargo, este uso se verá incrementado gracias a la aparición y el fomento de propuestas como el reconocimiento legal de la firma electrónica como equivalente a la firma manuscrita cuando ésta ha sido generada en un dispositivo seguro como puede ser una tarjeta inteligente (aprobado tanto en la directiva europea de firma electrónica [96] como en el modelo UNCITRAL sobre firma electrónica propuesto por Naciones Unidas [298]), el DNI electrónico o las tarjetas EMV como sustitutas de las tarjetas de crédito de banda magnética.

El principal problema de ambos tipos de propuestas (tanto basadas en tarjeta como las no basadas en tarjeta) es que no tienen en cuenta la seguridad a lo largo de todo de las principales fases del proceso de compra de productos electrónicos (negociación, pago y distribución), de forma que, se garantice el no repudio y un intercambio equitativo. Si se garantizan estas propiedades el cliente que realiza el pago de un producto puede estar seguro de que lo recibirá. Además, también hay que tener en cuenta que la distribución del producto tiene que ser realizada de forma segura evitando así que otros usuarios se puedan aprovechar del pago realizado por un determinado usuario.

Por tanto, a la vista de las conclusiones presentadas, podemos derivar que surge la necesidad de un sistema de pago que, a la misma vez que tiene en cuenta estos aspectos de seguridad, utilice las ventajas de la tarjeta inteligente como mecanismo de pago. Además, debería ser factible que la propuesta se pudiera aplicar a distintos escenarios y que estuviera diseñada para la adquisición eficiente de contenidos electrónicos.

2.3. Frameworks de pagos electrónicos B2C

En las anteriores secciones hemos presentado distintos protocolos de pago, cada uno con sus características particulares en cuanto a funcionalidad, seguridad, etc. Conforme pasa el tiempo van apareciendo más protocolos que intentan solventar los problemas que presentan las soluciones anteriores o tratan de cubrir nuevos escenarios o ámbitos hasta ahora no cubiertos. Por tanto, podemos pensar en la posibilidad de que a la hora de realizar un pago por un contenido en la Web dispondremos de varias opciones. Sin embargo, estas opciones de pago, en general, son incompatibles entre ellas.

En este escenario de múltiples soluciones a la hora de realizar un pago, se debería facilitar a clientes y vendedores la elección y el uso de un protocolo de entre las opciones disponibles. Así, a pesar de las diferencias que presentan los distintos protocolos, su uso para la realización de pagos debería ser sencillo, transparente y uniforme. Tal y como mencionan Gefen et al. en [118] y Wareham et al. en [309] se debería ofrecer un mecanismo uniforme que genere confianza en los usuarios. Es importante que para los usuarios, independientemente del protocolo utilizado, el proceso de pago siga los mismos pasos de forma que el usuario esté seguro de que está realizando el proceso de forma correcta. Para este propósito aparecen los frameworks de pago o servicios genéricos de pago.

Entendemos por *framework de pagos* un diseño o implementación que soporta distintos mecanismos de pagos y que es extensible. Así, un framework de pagos tiene por objetivo armonizar y ofrecer un entorno uniforme y estable para la elección y el uso de distintos sistemas de pago [165, 166]. En esta sección presentaremos las características así como las principales propuestas que han aparecido con respecto a los frameworks de pago.

2.3.1. Caracterización de los frameworks de pagos

Como acabamos de mencionar, la idea de proponer un framework de pagos o un servicio de pagos genéricos [165, 166] nace de la necesidad utilizar en las aplicaciones de comercio/negocio distintos sistemas de pago que son incompatibles entre sí. A la misma vez que facilita a clientes, vendedores y desarrolladores de pago la incorporación de nuevos sistemas de pago.

Cuando alguien quiere realizar un pago, a través de un framework de pagos, debería ser capaz de identificar los sistemas de pago que tiene en común con el receptor del pago y, entonces elegir uno de ellos para efectuar el pago. En concreto, las características que debería ofrecer un framework de pagos son:

- Soportar distintos métodos de pago (crédito, débito, etc) protocolos y mecanismos de pagos (por ejemplo, Millicent [117], D-Cash [148], Paycash [219], SET [274], etc).
- Permitir el uso de distintas entidades financieras (Visa, Mastercard, etc.) con los distintos protocolos.
- Facilitar que el pago se pueda realizar a distintas entidades en representación del vendedor y que denominaremos *proveedores de servicios de pago* (PSPs). En el caso

de que el vendedor soporte un determinado protocolo, diremos que para el protocolo el vendedor representará el rol del PSP.

- Posibilitar la especificación de distintos precios para las distintas combinaciones de protocolos, marcas financieras y PSPs. A un conjunto de estos elementos es lo que denominaremos *opciones de pago*.
- Facilitar la negociación del precio de las opciones de pago y del producto.
- El intercambio sobre las distintas opciones de pago junto con sus precios debería de ser intercambiado de una forma segura, garantizando al menos la confidencialidad, integridad y autenticidad.
- Permitir el soporte de distintos modelos de negocio: pago por uso, pago por tiempo, pago por volumen de datos, que un pago de derecho al acceso de un conjunto de recursos, etc. Para el soportar algunos de estos modelos será necesario la utilización de sesiones.
- Permitir la elección del protocolo de pago a utilizar entre los distintos soportados por el cliente y vendedor.
- Ser extensible y que permita incorporar nuevos protocolos que surjan.
- La incorporación de nuevos mecanismos y protocolos de pago debería ser a través de una interfaz de programación de aplicaciones (Application Programming Interface – API) claramente definida y que facilite la integración de éstos a clientes y vendedores. El API ocultará las particularidades de cada protocolo. Así, esta API nos permitirá crear *wallets* – monederos que gestionan la información relacionada con un determinado(s) método(s) de pago – que soporten el framework. Facilitando así el trabajo a los desarrolladores de sistemas de pago. Es importante comentar que aquí el término monedero se emplea de una forma más genérica a como se utiliza en la sección 2.2, donde este término se refiere a una tarjeta inteligente que implementa un sistema de pagos. Para evitar esa confusión en esta sección utilizaremos el término *wallet*.
- Orientado al cliente para evitar posibles ataques que implican la recepción de solicitudes de pagos en el wallet de cliente.
- Encapsulación de los mensajes del protocolo de pago elegido para la transacción en un protocolo de pagos genéricos que facilite su uso en diferentes escenarios de pago.
- El protocolo debería facilitar como ya hemos mencionado antes el uso de sesiones así como la posibilidad de realizar diversos pagos con el mismo vendedor de forma consecutiva eficientemente (lo que denominaremos *pagos sucesivos*) [81].
- Este framework debe facilitar el uso de los sistemas de pago en las aplicaciones.

Si además, este framework lo queremos utilizar para la realización de pagos por productos que se pueden adquirir en la Web, éste además debería soportar la especificación de los precios y las distintas opciones de pago soportadas para pagar por un determinado producto en la Web. Esta información se debería expresar de la forma más genérica posible y debería ir asociada al *enlace de pago* (enlaces en los que es necesario efectuar un pago antes de poder acceder a su contenido) que se corresponde con el

producto. Finalmente, para incluir esta información e incluir los elementos necesarios en el framework, la solución debería de estar basada en los mecanismos de extensibilidad existentes en los distintos elementos Web: HTML, HTTP, mecanismos de extensibilidad de los navegadores, etc. El objetivo es facilitar la incorporación de este framework a los sistemas Web ya establecidos. Todas estas características, como posteriormente veremos, serán tenidas en cuenta en las propuestas que presentamos en el Capítulo 4 para ofrecer una solución global que cubra todas las necesidades planteadas.

2.3.2. Principales propuestas relacionadas con frameworks de pagos

En esta sección presentaremos las principales propuestas de frameworks de pago que tratar de cubrir los requisitos mencionados en la sección anterior.

JEPI

JEPI (Joint Electronic Payment Initiative) [56] fue una propuesta conjunta del W3C junto con el consorcio CommerceNet [60] realizada en el año 1997. El objetivo de JEPI era automatizar la negociación del protocolo de pago que sea más adecuado, para cliente y vendedor, en una transacción donde ambos soportan distintos protocolos. Así, el diseño de JEPI pretende actuar como un sistema general de forma que facilite y mejore la interoperabilidad, permite a los usuarios elegir el mecanismo de pago a utilizar y les presenta la información de pago de una manera uniforme [56].

JEPI define dos nuevos protocolos sobre HTTP para llevar a cabo la negociación y la selección del mecanismo de pago a utilizar: PEP y UPP (ver Figura 2.7).

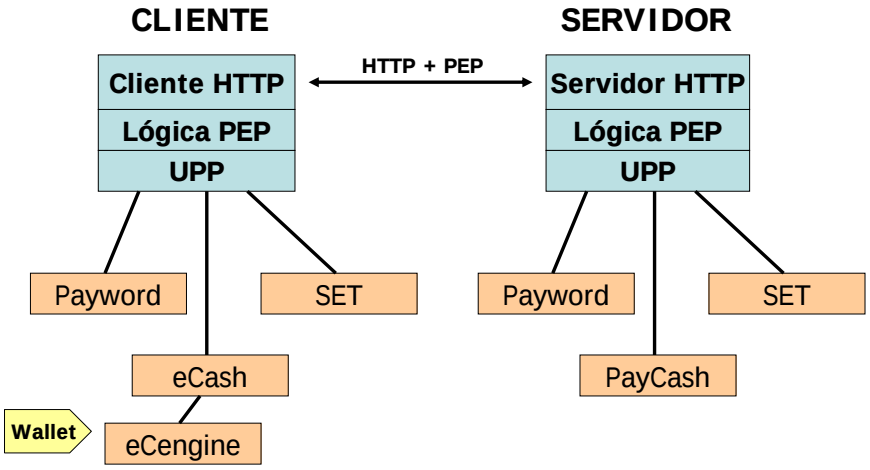


Figura 2.7. Arquitectura de JEPI.

PEP (Protocol Extension Protocol), que actúa como capa de nivel inferior, es un protocolo de negociación genérico que permite a un cliente y a un servidor Web negociar las extensiones que cada uno posee dentro de HTTP. Las extensiones de PEP incorporan nuevas características a HTTP y están basadas en el uso de nuevos campos de cabecera

para contener los identificadores de las extensiones e información relacionada sobre los clientes HTTP. PEP fue utilizado también en el proyecto PICS (Platform for Internet Content Selection) para el intercambio de metadatos asociados a contenido disponible en Internet. En JEPI, PEP es la base para UPP, que es el encargado de llevar a cabo la negociación del mecanismo pago.

UPP (Universal Payment Preamble) [80] es el núcleo de JEPI, actuando como capa de nivel superior y consiste en un módulo de extensión específico que se utiliza para negociar los instrumentos de pago (cheque, tarjeta de crédito, dinero electrónico, etc.), las marcas financieras (VISA, MasterCard, American Express, etc.) y el protocolo de pago (SET, Millicent, etc.). Cada sistema de pago se considera una extensión PEP y se identifica por medio de una URL. Además, en JEPI se define un mecanismo de inicio del sistema de pago elegido.

En este framework no se define un mecanismo para intercambiar de forma uniforme los mensajes de los protocolos de pago, sino que se deja abierto a la implementación que se lleve a cabo de los wallets para el cliente y el servidor. Por tanto, dos implementaciones de distintos vendedores o desarrolladores de software de pago podrían no interoperar. Aunque no se soporta este aspecto, sí se contemplaba que en las posibles líneas futuras se extendiera PEP para contener los mensajes específicos del protocolo. Además, en esta propuesta no se tiene en cuenta que la información de pago se envíe de una forma segura, ya que se intercambia sobre HTTP. Este intercambio es independiente de la seguridad del protocolo elegido para realizar el pago. Para obtener un mayor nivel de seguridad, se debería utilizar un canal SSL/TLS, opcionalmente combinado con el uso de firma electrónica, para que un tercero no modificase, sin que las partes se diesen cuenta, la información durante el intercambio, tal y como se presentará en nuestra propuesta del Capítulo 4. Además, se debería tener en cuenta, posibles ataques de man-in-the-middle, por lo que habría que definir qué servidores se consideran de confianza.

En las posibles líneas futuras de PEP también se planteaba la posibilidad de permitir diferentes modelos de pago y/o modos de operación. Finalmente, en esta propuesta, no definen cómo incluir la información de pago en la Web.

A pesar de las expectativas que se crearon en el proyecto, éste no continuó desarrollándose. Las posibles razones son varias:

- Algunos de los problemas planteados anteriormente.
- Falta de confianza de los usuarios en los sistemas de pago.
- Los protocolos de pago en aquel momento no eran suficientemente seguros o presentaban problemas de usabilidad para los usuarios.

Aunque el proyecto no continuo, las ideas propuestas han servido de base para otras propuestas que comentaremos a continuación como SEMPER, una propuesta de pagos del W3C, IOTP, así como en la propuesta que presentamos del framework de pagos por clic del Capítulo 4.

SEMPER

SEMPER (Secure Electronic Marketplace for Europe) [165, 273] era un proyecto parcialmente subvencionado por la Comisión Europea y formado por un consorcio de alrededor de 20 compañías, instituciones financieras, universidades europeas e institutos tecnológicos/de investigación. Este proyecto desarrollado entre los años 1995 y 2000 pretendía trabajar en distintos temas relacionados con el comercio electrónico sobre redes inseguras, tales como aspectos legales, sociales o técnicos.

Los objetivos que persigue SEMPER [165] son definir una arquitectura abierta que sea independiente de la arquitectura de red, del sistema operativo y del software utilizado, de tal forma que pueda evolucionar sin problemas con la inclusión de nuevos componentes por medio de plug-ins. Esta arquitectura está dividida en cuatro capas: la capa de negocio, la capa de intercambio, la capa de transferencia y la capa de servicios de soporte. Dentro de estos objetivos, uno de los más importantes era diseñar e implementar un servicio de pago genérico que proporcione un framework que permita a las aplicaciones de negocio ocultar las particularidades de los distintos métodos de pago a través de APIs, dándole al usuario la posibilidad de llevar a cabo el pago de forma transparente [166, 220].

El principal componente de este servicio es una jerarquía de APIs para la transferencia de valor monetario. La jerarquía de la API refleja la separación de acuerdo a los modelos de pago que fueron comentados en la Sección 2.2.1. Así, esta jerarquía consiste en un API raíz, común a todos los modelos de pago, y en extensiones específicas para cada modelo. Los modelos en los que se divide la API son en instrumentos de pago, tales como el dinero electrónico, y en sistemas basados en cuenta.

Además de estas interfaces, este servicio genérico ofrece otras características como son mecanismos de selección del instrumento de pago a utilizar en la transacción, servicios de información, control y gestión (que posibilitan el desarrollo de aplicaciones que usan el servicio de pago genérico), herramientas y el framework necesario para incorporar los sistemas de pago actuales en el servicio de pago genérico.

Cada sistema de pagos específico implementa alguna de las APIs anteriormente mencionadas. Los servicios que proporciona un wallet son: crear una instancia de ese monedero, configuración, inicialización, creación de transacciones y provisión de servicios de información.

En la arquitectura existe un elemento denominado *gestor del pago* que lleva a cabo dos funciones. Por un lado, elige el wallet más adecuado para una determinada transacción. Por otro lado, lleva un registro de los wallets disponibles así como de la información relacionada con las transacciones activas y pasadas. Como podemos ver en la Figura 2.8, el controlador del gestor del pago utiliza los distintos wallets que, a su vez, se comunican con la instancia del sistema de pago elegido a través de adaptadores especializados. Cada adaptador proporciona la unión entre un modelo de pago y la instancia externa del sistema de pago.

El principal punto fuerte de esta propuesta es la definición del wallet a partir de la generalización de las transacciones a los tipos que acabamos de comentar. Sin embargo, este modelo está más orientado a aplicaciones que a los entornos Web, por lo que no

define la información de pago asociada a los enlaces, ni tampoco define como encapsular los modelos de pago. De esta forma, una vez que el wallet inicia la transacción, éste es el responsable del intercambio de los mensajes con el wallet del servidor. Este problema veremos cómo se resuelve por medio de la propuesta de wallet que proponemos en el Capítulo 4.

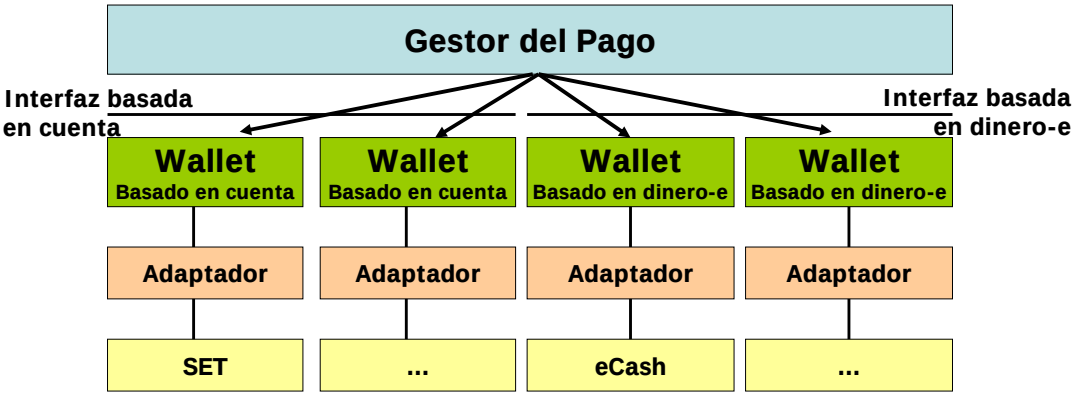


Figura 2.8. Componentes de la arquitectura de pagos de SEMPER.

El proyecto concluyó en el año 2000. Los resultados se publicaron en [165] y en el portal Web del proyecto [273] está disponible aún un demostrador. Además, este framework de pagos fue adoptado por el servidor de pagos de IBM (IBM payment server) y a día de hoy se siguen utilizando algunos de estos componentes en sus aplicaciones. Aunque ahora estas aplicaciones están más orientadas al entorno de pagos bancarios debido a la escasa adopción que tuvieron los sistemas de pago cuando se lanzó esta iniciativa. Estas ideas también han sido de utilidad para IOTP o para otros modelos como el de los cupones electrónicos [109, 286] así como para las propuestas que presentamos en el Capítulo 4.

W3C Common Markup for micropayment per-fee-links

El grupo de trabajo de etiquetado de micropagos (*Micropayment Markup Working Group*) del W3C en 1999 propuso la especificación denominada Etiquetado Común para Enlaces de Pago con Micropagos (*Common Markup for micropayment per-fee-links*) [193]. El objetivo de esta especificación es proporcionar una forma extensible de incluir, en una página Web, toda la información necesaria para inicializar un sistema de micropago (importes y divisas, sistemas de pago, etc.) cuando se requiere realizar un pago por acceder a un contenido Web. Al incluir esta información en la página Web, se pretende facilitar que diferentes monederos electrónicos de micropagos coexistan de una forma interoperable.

Esta propuesta es la primera en introducir el concepto de *enlace de pago*. El objetivo de estos enlaces es facilitar que el proceso de compra de un producto electrónico (página Web, imagen, etc.) sea lo más sencillo posible: tan simple como efectuar un clic en un enlace Web.

La especificación se centra en definir cómo incluir la información de los enlaces de pago en la página HTML que el vendedor envía al navegador del cliente. Aunque no define el mecanismo de transporte de la página HTML, en general, será HTTP o HTTPS. Para incluir la información de pago en los enlaces se definen una serie de elementos como el precio, el texto que describe lo que el cliente solicita con el enlace, la imagen gráfica de lo se solicita con el enlace, los sistemas de pago a utilizar, el identificador del vendedor y su nombre, cuándo expira la información de pago, etc. La lista completa de los distintos elementos que define la especificación se encuentra en [193]. Esta información se puede incluir por medio de Applets Java o ActiveX, o mediante RDF [19]. Esta propuesta tampoco define cómo se produce el envío de los mensajes de pago. Estos problemas serán resueltos mediante la redefinición de la información a incluir para expresar la información de pago que presentamos en el Capítulo 4.

Para el manejo de esta información en el cliente se propone el uso de un módulo llamado *manejador de enlaces de pago*. Éste, a su vez, está conectado con varios wallets que implementan los protocolos de pagos específicos. Sin embargo, no se proporciona la API entre el manejador y los wallets. Así como tampoco se define como se llevará a cabo el transporte de los mensajes de los protocolos de pago.

Algunas empresas que implementan sistemas de micropagos, como Cartio [43] o NewGenPay [203], están ofreciendo contenidos basados en esta especificación.

IOTP

Internet Open Trading Protocol (IOTP – Protocolo de Comercio Abierto para Internet) [32, 33, 82] es un protocolo que nació en el año 2000 y fue desarrollado por el Internet Engineering Task Force (IETF), con la idea de convertirse en un estándar internacional para comerciar en Internet. En concreto, el objetivo era desarrollar un protocolo de compras para el modelo B2C que estuviera centrado, sobre todo, en el pago y en la entrega, y que se convirtiera en un estándar que ayudase a la extensión de los mecanismos de pago.

IOTP define los siguientes roles (*trading roles*) que pueden participar en una transacción de comercio:

- *Consumidor* o *cliente* que es quien paga por y recibe bienes y servicios.
- *Vendedor* que es a quien se le realiza la compra y el responsable de proporcionar los bienes y servicios por los que es pagado.
- *Manejador o gestor del pago* que acepta los pagos del cliente en representación del vendedor.
- *Manejador o gestor de la entrega* que es quien proporciona los bienes o servicios del vendedor.
- *Proveedor de atención al cliente del vendedor* que es quien resuelve los problemas y disputas del consumidor.

Aunque se realiza esta distribución entre los roles de vendedor, gestor del pago, gestor de la entrega y proveedor de atención al cliente del vendedor, éstos podrían ser desempeñados por una única entidad u organización (por ejemplo, el vendedor).

IOTP define las siguientes transacciones básicas (*IOTP transactions*):

- *Compra* que implica una oferta, un pago, y, opcionalmente, una entrega.
- *Devolución* que soporta que un pago que anteriormente se llevó a cabo se pueda devolver.
- *Intercambio de valor* que implica dos pagos como resultado del intercambio de valor.
- *Autenticación* que permite que una parte se autentique frente a otra para que esté segura de con quién está realizando la transacción.
- *Retirada de dinero electrónico* desde una entidad financiera.
- *Depósito de dinero electrónico* a una entidad financiera.
- *Información* para consultar el estado de una transacción IOTP que está en progreso o completada.
- *Ping* que permite, a una aplicación IOTP, comprobar si otra aplicación IOTP está funcionando.

Cada transacción implica la participación de una o más entidades desempeñando alguno de los roles de comercio que hemos mencionado anteriormente, así como un conjunto de intercambios de comercio (*trading exchanges*) que implican el envío y recepción de datos entre los distintos roles de comercio en forma de un conjunto de componentes de negocio (*trading components*). Estos componentes de negocio se agrupan en bloques y, finalmente, se envían un conjunto de bloques en un mensaje IOTP. La realización de una transacción puede implicar el intercambio de varios mensajes.

En IOTP se definen cuatro intercambios de comercio (*trading exchanges*):

- *Oferta* en la que se define la razón por la que cliente y vendedor realizan un intercambio comercial.
- *Pago* en la que se produce un pago del cliente al manejador del pago, o viceversa.
- *Entrega* en la que se proporcionan los bienes electrónicos o físicos por parte del manejador de la entrega.
- *Autenticación* en la que cualquier rol de comercio autenticar a cualquier otro rol de comercio para asegurarse de que es quién dice ser.

Las transacciones de IOTP consisten en la combinación de varios de estos intercambios de comercio. Por ejemplo, una transacción IOTP de compra incluye intercambios de comercio de oferta, pago y entrega.

Los mensajes de IOTP están definidos por medio de XML e incluyen la posibilidad de que estén firmados conforme a un formato definido en [70], aunque no conforme a XML Signature [83, 85], ya que en su momento utilizaron su propio formato de firma electrónica. Así, es posible garantizar la integridad y autenticidad de los mensajes intercambiados. El uso de la firma es opcional ya que IOTP se puede utilizar con distintos

protocolos de pagos, como los de micropagos y, en este caso, podría no requerirse. De todas formas, aunque el protocolo sea seguro también puede requerirse la firma ya que, previo al proceso de pago, se intercambia cierta información relacionada con éste que podría ser interceptada y modificada por un tercero.

Una descripción más detallada de los distintos componentes que forman parte de las transacciones, intercambios, bloques, mensajes, gestión de errores, análisis de seguridad, etc. se puede encontrar en [32, 33].

El principal aspecto a destacar de IOTP es que el proceso de pago se diseñó para que fuera independiente del protocolo de pagos a utilizar y, por tanto, se pudiera utilizar con protocolos como SET, eCash, Payword, etc. El proceso de pago que se seguiría conforme a IOTP aparece reflejado en la Figura 2.9.

En este proceso de pago, el primer mensaje que el cliente envía al vendedor contiene una cabecera MIME que provocará que lance la aplicación que soporta IOTP. La aplicación IOTP se divide en tres componentes principales: el *núcleo de la aplicación IOTP*, que procesa las partes genéricas del protocolo; el *software de pago existente* que procesa las peticiones de un determinado protocolo de pago; y el *middleware de IOTP*, que actúa de enlace entre los dos componentes anteriores, de forma que encapsula el software de pago existente con una API de pago [26] que será invocada por el núcleo de la aplicación IOTP. Esta API de pago considera los siguientes tipos de transacción IOTP: compra, devolución, intercambio de valor, retirada y consulta.

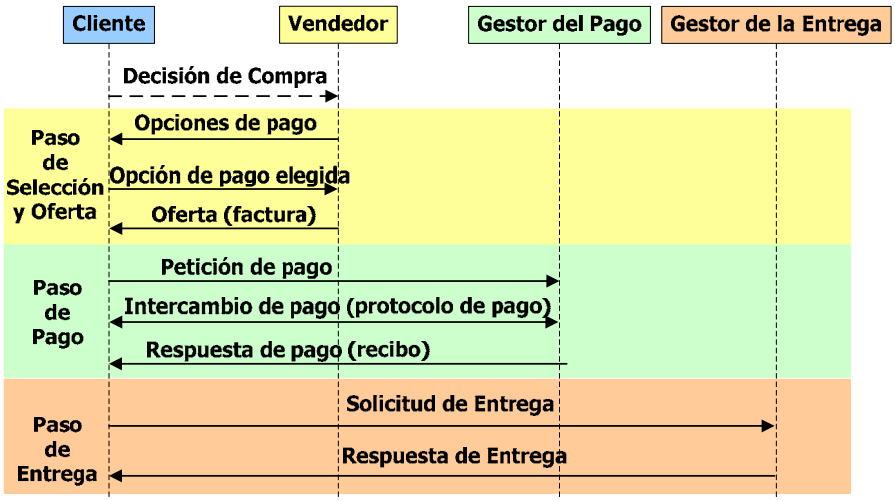


Figura 2.9. Flujo de mensajes de IOTP.

Esta es la única de las propuestas analizadas hasta ahora que tiene en cuenta el transporte de los mensajes de pago del protocolo elegido. Otros aspectos positivos a destacar es que permite la especificación de diferentes precios para cada una de las opciones de pago así como ha definido una API de wallet genérica. Estos aspectos también han tenidos en cuenta para el diseño del protocolo EPP que presentamos en el Capítulo 4.

Sin embargo, hay diversos aspectos que el diseño de IOTP no tiene en cuenta y que hacen que esta propuesta no sea lo suficientemente genérica ni completa para los objetivos marcados de un framework de pagos para la Web. Algunos de estas deficiencias que mencionaremos a continuación fueron propuestas por los propios autores de IOTP como requisitos para una futura versión [33, 81].

En IOTP la solicitud de oferta no está codificada como mensaje que forme parte del protocolo, por lo que no se puede incluir ni autenticación, ni firma, ni parámetros de la transacción. El protocolo tampoco soporta la negociación del precio del producto. Otro aspecto, que es fácilmente solucionable, es la sustitución de la firma XML definida en IOTP por el uso de la firma XML Digital Signature (XMLDsig) [84, 85] conforme al IETF/W3C.

Para la futura versión también se propone que sea desarrollada como parte de una filosofía de diseño Web más amplia y descentralizada, con URIs, datos en la Web y extensibilidad. Hay que tener en cuenta además, que en IOTP tampoco se definen es los enlaces de pagos.

Finalmente, otra necesidad que se plantea es la de soportar pagos repetidos/futuros, de forma que un pago no sólo cubra la actual compra si no futuras compras que se producirán en un momento dado. Por ejemplo, que un pago nos de acceso a varios artículos. Estos aspectos, como posteriormente veremos, están contemplados dentro del protocolo EPP presentado en el Capítulo 4.

IOTP fue implementado por en Japón por varios bancos. Actualmente, el grupo de trabajo de IOTP en el IETF está cerrado, aunque hay algunas de las propuestas relacionadas que siguen desarrollándose. Con respecto a IOTP, desde 2002 en el que se definieron los requisitos para la nueva versión no se ha vuelto a publicar por parte del IETF ningún draft o actualización de este protocolo. Aparte de las deficiencias encontradas, otro motivo por el que puede que esta propuesta no alcanzara la repercusión esperada se deba al hecho de que coincidiera con los años en los que se produjo la explosión de la burbuja de las *.com* que hizo que en esos años el comercio electrónico disminuyese de forma significativa [18]. Otro motivo adicional es la falta de confianza mostrada hasta ahora por los usuarios ante la seguridad de los métodos de pago en Internet. A pesar de este hecho, el protocolo continúa estudiándose y aplicándose a diversos escenarios, sobre todo relacionados con los derechos de autor [121, 161]. Quizá la actividad del IETF sobre este protocolo puede volver de nuevo resurgir ya que de nuevo el comercio electrónico vuelve a tener un futuro prometedor tal y como se menciona en [173].

Prototipo de sistema de comercio electrónico

Meng y Zhang [190], en 2005, proponen un prototipo de sistema de comercio electrónico basado en tecnología Web. Los elementos que forman parte de esta propuesta son el sistema del vendedor, un wallet de cliente que se encuentra alojado en un servidor y una pasarela de pago. En esta propuesta, el wallet del cliente alojado en un servidor soporta distintos protocolos e instrumentos de pagos y es extensible. El cliente, para comunicarse con su wallet utiliza el protocolo SSL/TLS y servicios Web seguros [200]. Los principales componentes del wallet (ver Figura 2.10) son el *gestor de instrumentos de pago*, que

permite el uso de distintos instrumentos tales como dinero electrónico o tarjetas de crédito, y el *gestor de protocolos de pagos* que permite utilizar distintos protocolos de pago con distintos instrumentos de pago. Otros componentes que se han definido, han sido los módulos necesarios para la autenticación del usuario, los módulos relacionados con las transacciones y los módulos de información. La comunicación de este wallet con el vendedor y con la pasarela de pagos se lleva a cabo por medio de servicios Web seguros. La principal ventaja del uso del wallet en el servidor es que el cliente puede utilizar su wallet en cualquier ordenador [65].

Este sistema requiere, por tanto, que, para cada transacción de pago, participe una tercera entidad (el wallet alojado en el servidor), lo cuál lleva a que el tiempo necesario para realizar la transacción sea mayor (salvo que quien almacene el wallet de servidor sea el mismo vendedor al que se efectúa el pago) [65]. Además, el cliente tiene que depositar la confianza en esta tercera entidad, ya que ésta es la que gestiona toda su información de pago. El sistema tampoco define cómo llevar a cabo la elección del protocolo de pago, ni la negociación del precio, ni los enlaces de pago, ni el encapsulado de los mensajes de pago en un protocolo genérico. En este caso, los mensajes se encapsulan en mensajes SOAP que se intercambian entre los distintos elementos.

Esta propuesta académica, que sepamos, hasta ahora no ha sido adoptada por la industria de las tecnologías de Internet y medios de pago.

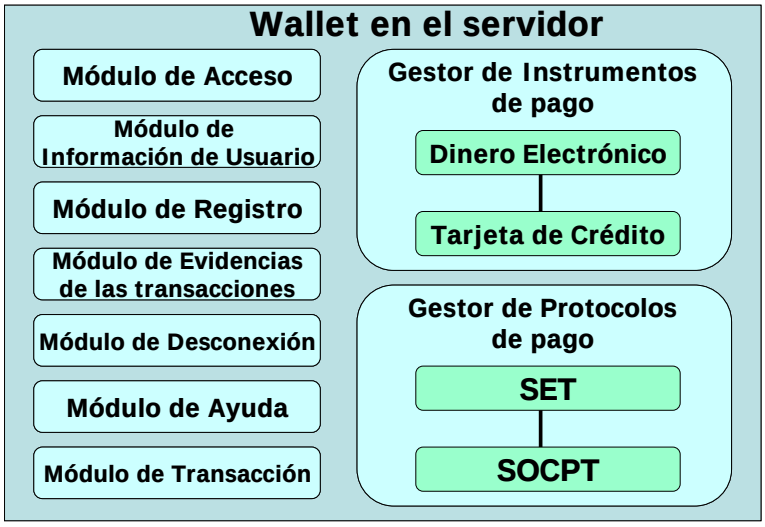


Figura 2.10. Arquitectura de la propuesta de Meng y Zhang.

2.3.3. Comparativa de los frameworks de pagos electrónicos B2C

En esta sección presentamos una tabla (ver Tabla 2.3) en la que resumimos las distintas características ofrecidas por cada una de las propuestas mencionadas a lo largo de las anteriores secciones con el fin de poder compararlas fácilmente. Esta tabla está basada en las características definidas en la Sección 2.3.1. De esta comparativa intentaremos extraer conclusiones que nos permitan discernir cuáles son los puntos fuertes y débiles de las

propuestas aparecidas hasta la fecha. Nuestro objetivo es poder determinar cuáles son los distintos aspectos que no cubren estas propuestas para ofrecer un framework genérico de pagos para la Web. Éste debe definir la información a incorporar en la Web, soportar distintas opciones de pago, distintos precios, permitir la negociación del precio, ofrecer un protocolo genérico de intercambio de mensajes que garantice la seguridad de la información intercambiada y ser extensible.

Propuesta / Propiedades	JEPI	SEMPER	W3C	IOTP	Meng
Selección del método de pago	x	x	x	x	x
Distintos precios por método de pago		x	x	x	
Negociación de precios		x			
Orientado al cliente	x	x	x	x	
Protocolo de pagos genérico				x	
Ofrece sesión de pago					
Permite soportar pagos sucesivos					
Soporta distintos modelos de negocio					
API de wallet genérica	x_0	x	x_1	x	x
Enlace de pago			x		x
Intercambio de información segura				x_2	
Requiere re-implementar navegador y servidor Web	x				

Tabla 2.3. Comparativa de las propuestas relacionadas con frameworks de pagos.

Notas:

x_0 – Esta característica la proponen como una tarea a realizar en el futuro.

x_1 – Información no disponible, aunque se supone que debería serlo.

x_2 – A decisión de los participantes involucrados en la transacción.

De la comparativa podemos destacar que una de las características en la que parecen coincidir todas las propuestas es en la necesidad de permitir la elección al cliente del método de pago a utilizar. De los sistemas mostrados, la mayoría coincide en mostrar que el precio puede variar dependiendo del sistema de pago utilizado. También, la mayoría de ellas parece decantarse porque en el framework, el proceso de pago tiene que estar orientado al cliente (para evitar posibles ataques que implican la recepción de solicitudes

de pagos en el wallet de cliente) y debe definir una API genérica que abstraiga el funcionamiento de los distintos protocolos de pago.

Sin embargo, aunque cada una de las propuestas cubre distintos aspectos que se mencionaron en la caracterización de los frameworks de pagos, ninguna satisface todos los requisitos que debería soportar el framework. En este sentido, los principales aspectos que deberían abarcar serían la definición de los enlaces de pago, el soporte para pagos sucesivos y múltiples modelos de negocio. Otro aspecto a tener en cuenta es la API, ya que, aunque algunas tienen una interfaz genérica para el proceso de pago, los métodos de algunas de ellas no son lo suficientemente genéricos y algunas veces dependen del tipo de instrumento utilizado, como en el caso de SEMPER.

2.3.4. Conclusiones

A lo largo de la Sección 2.3 hemos presentado los frameworks de pago para entornos B2C. Hemos visto que éstos surgen de la necesidad de soportar el uso de múltiples sistemas de pagos de una forma uniforme. Una vez introducidos hemos presentado las características que deberían satisfacer y las principales propuestas que han surgido.

Las propuestas analizadas hemos visto que presentan determinadas deficiencias que hacen que las soluciones propuestas hasta ahora no sean lo suficientemente genéricas o no comprendan todos los componentes que se requieren. Sin embargo, existen otros factores adicionales que han hecho que estas propuestas no han sido adoptadas como la falta de confianza en los sistemas de pagos tal y como mencionábamos en la Sección 2.2 así como que el proceso de pago con estos frameworks no sea lo suficientemente uniforme para el soporte de pagos en la Web. En respuesta a esta problemática, en primer lugar, se deberían solucionar los problemas relacionados con los sistemas de pagos para que se genere confianza en los sistemas de pago. En segundo lugar y en paralelo, se deberían cubrir los problemas que presentan estos frameworks y que hacen que los usuarios no vean el pago con distintos sistemas de pago de un modo uniforme y que les inspire confianza y les permita estar seguros de que están realizando el proceso de pago correctamente.

2.4. Conclusiones del capítulo

En los últimos años el uso de las nuevas tecnologías y la creación de contenidos en la Web han obtenido un gran auge. Una de las principales áreas que se han beneficiado de este auge ha sido el comercio electrónico, en particular el comercio B2C. En él, los distintos creadores, editores, productores y distribuidores de contenido electrónico quieren obtener beneficios de los contenidos que crean, editan, producen y distribuyen respectivamente. Con el fin de que los vendedores puedan cobrar por sus productos y servicios aparecen los protocolos de pago.

En este capítulo hemos analizado las características que deben satisfacer los protocolos de pago cuando queremos utilizarlos para la adquisición de productos electrónicos en la Web. Una vez introducidas estas características hemos analizado los principales protocolos de pago con respecto a estas características. Entre estas caracterísitsticas destacan la seguridad del método de pago que es uno de los aspectos que ha provocado desconfianza

en los usuarios y que ha hecho que éstos no estén tan extendidos como se esperaba. En particular, para el caso de los productos electrónicos es importante que se garantice el no repudio y el intercambio equitativo. Así tanto el usuario como el vendedor van a estar seguros que de uno recibirá el producto y otro el pago. En caso contrario, ninguno de los dos obtendrá ninguno de estos elementos. Además, otro aspecto a cubrir por en estos protocolos es que no sólo el pago se realice de forma segura, sino que también las fases de negociación y entrega del producto se lleven a cabo de forma segura.

También hemos visto como las tarjetas inteligentes pueden ser un elemento fundamental a la hora de incrementar la seguridad del método de pago, facilitar la movilidad y así incrementar la confianza del usuario. Sin embargo, tal y como hemos visto la mayoría de estas soluciones no están diseñadas para su uso en Internet de forma eficaz. El principal motivo se debe a que el vendedor tiene que integrar dispositivos con módulos SAM que permitan realizar el pago con la tarjeta. Este pago requiere el intercambio de varios mensajes entre la tarjeta y el SAM lo que puede llevar a que el pago se haga de forma lenta. Otro problema menor que presentaban las tarjetas inteligentes es que su uso hasta ahora no estaba muy extendido.

Por tanto, con respecto a los protocolos de pago podemos concluir que surge la necesidad de plantear protocolos de pago que permitan cubrir las fases de negociación, pago y distribución del contenido electrónico de forma segura. Estos protocolos deberían estar basados en el uso de monederos electrónicos que aportan más seguridad y que facilitan la movilidad del usuario. Además, estas soluciones de pago basadas en tarjetas deberían facilitar su uso a los vendedores e intentar que el proceso de pago simplificara la utilización de los dispositivos con módulos SAM.

En este capítulo también hemos visto que otro de los problemas que han llevado a que los sistemas de pago no estén tan extendidos es el uso de frameworks de pagos. Es decir, una serie de mecanismos uniformes para el uso de distintas soluciones y protocolos de pago. En el capítulo hemos analizado las características que éstos deben proporcionar para ser utilizados en la Web para los principales tipos de productos y servicios (contenidos y servicios Web y, contenidos y servicios basados en sesiones multimedia). También hemos presentado las principales propuestas y las hemos analizado con respecto a las características mencionadas. Del análisis se desprende que no existe una solución genérica que cubra todos los aspectos necesarios para una propuesta de este tipo. Por tanto, es necesario la definición de frameworks de pago B2C que definan la información a incorporar en la Web, soporten distintas opciones de pago, distintos precios, permitan la negociación del precio, ofrezcan un protocolo genérico de intercambio de mensajes que garanticen la seguridad de la información intercambiada y sea extensible.

A la vista de los resultados de los análisis realizados sobre los sistemas y frameworks de pagos podemos también concluir que, aún a pesar de que este campo se ha realizado un exhaustivo trabajo de investigación, todavía quedan determinados aspectos que deben ser cubiertos para facilitar su extensión y el uso de los sistemas de pagos en diferentes escenarios.

En los siguientes capítulos de esta tesis veremos cómo cubrir las necesidades que acabamos de mencionar, tanto desde el punto de vista de los protocolos de pago como de los frameworks necesarios para su uso.

Capítulo 3

Propuestas de Sistemas de Pago B2C basadas en Monedero Electrónico

En el capítulo anterior presentamos todos los conceptos y propuestas relacionadas con los sistemas de pago, las tarjetas inteligentes y los monederos electrónicos. En este capítulo pasaremos a describir dos propuestas que solucionan los problemas planteados durante dicho análisis.

El objetivo de nuestras propuestas es proporcionar sistemas de pago basados en monedero electrónico que puedan ser utilizados en distintos ámbitos: máquinas de vending e Internet. Actualmente, el principal uso de los monederos ha sido en entornos de vending. Sin embargo, su uso en Internet es prácticamente inexistente. Estas soluciones pretenden facilitar a los vendedores la incorporación de sistemas de pago basados en monedero electrónico que pueden aportar mayor seguridad al sistema, mayor movilidad y su uso en distintos escenarios como los que acabamos de mencionar.

Nuestra primera propuesta de pago basada en tarjeta inteligente es el protocolo SPEED. Este protocolo permite el pago de productos electrónicos mediante el uso de un monedero electrónico como WG10 o CEPS. En SPEED con el fin de facilitar el soporte de este tipo de pagos a los vendedores, sin necesidad de instalar módulos SAM, y para garantizar la seguridad del sistema se introduce un tercero de confianza. Con esta propuesta facilitamos el ofrecer soluciones de pago basadas en monederos ya existentes y que utilizan estos tipos de dispositivos.

Este capítulo finaliza con la descripción de nuestra segunda propuesta, PURSE-COIN, que tiene por objetivo ofrecer un nuevo monedero electrónico que facilita la realización de pagos en la Web. Este monedero para efectuar pagos, en lugar de intercambiar mensajes con un módulo SAM, genera monedas electrónicas. De esta forma se elimina la necesidad de dispositivos con módulo SAM para realizar el cobro con un monedero. Por tanto, el vendedor puede recibir pagos de monederos de forma sencilla y sin necesidad de contactar con una tercera parte durante la fase de pago.

3.1. Introducción

Como se puso de manifiesto en el anterior capítulo, los principales protocolos de pago B2C para la compra de productos y/o servicios electrónicos presentan varios problemas. Dichos problemas son: no se tiene en cuenta la seguridad en todas las fases de la compra del producto (negociación, pago y distribución), no garantizan el no repudio, el intercambio equitativo y la atomicidad. La seguridad de los protocolos de pago es un aspecto fundamental de cara a generar confianza en los usuarios de los sistemas de pago. De las propuestas analizadas la más adecuada para este propósito era NetBill. Sin embargo, como se analizó esta propuesta requiere que el vendedor utilice certificados y que instale un sistema basado en Kerberos, lo que puede limitar la implantación de este tipo de solución. Además, la mayoría de los mensajes del protocolo y las operaciones relacionadas con éste las lleva a cabo el vendedor lo que puede limitar la escalabilidad de la solución. También veíamos que se ha encontrado un ataque a la propiedad de atomicidad. Por tanto, sería necesario proporcionar un protocolo que solucione estos problemas.

En este contexto, las tarjetas inteligentes y en particular los monederos electrónicos pueden aportar ventajas adicionales a un sistema de pagos ya que son más seguras que las tarjetas de banda magnética, pudiendo gestionar de forma segura la información criptográfica y de pago necesaria para realizar una transacción. Su incorporación a un sistema de comercio además facilitaría la movilidad del usuario a la hora de realizar pagos así como permitiría al usuario poder utilizar su tarjeta en distintos escenarios. Así el usuario podría utilizarla tanto en escenarios de vending o punto de venta como para pagos en Internet. De esta forma podría incrementar la confianza del usuario en los sistemas de pago al mostrarle seguridad y comodidad en el proceso. El principal problema de los monederos cuando se quieren utilizar para efectuar pagos en Internet es que suponen que el vendedor tenga que integrar dispositivos con módulos SAM en sus aplicaciones de comercio. Esta integración no es sencilla y además puede suponer que el proceso de pago sea lento debido a los mensajes que se tienen que intercambiar entre la tarjeta y el módulo SAM. Por tanto, se deberían proporcionar soluciones a estos problemas.

En respuesta a los problemas tanto de los protocolos de pago B2C como de los monederos proporcionamos una solución conjunta. Así en este capítulo presentamos dos soluciones de pago basadas en monedero que combinan las ventajas de los protocolos de pago B2C con los monederos a la misma vez que solucionan los problemas ya planteados.

Estas propuestas solucionan el problema de forma gradual y complementaria. En primer lugar, presentaremos el protocolo de pago SPEED (Smartcard-based Payment with Encrypted Electronic Delivery) [249] que garantiza las características de seguridad antes mencionadas en todas las fases de la compra. Esta propuesta además permite realizar el pago con alguno de los monederos existentes sin que el vendedor tenga que incorporar en sus soluciones de pago un dispositivo con un módulo SAM. Para resolver este problema, introducimos un tercero de confianza que es el encargado de garantizar el intercambio equitativo y la atomicidad así como de recibir el pago basado en monederos. La segunda propuesta PURSE-COIN presenta un nuevo monedero electrónico que simplifica la recepción de pagos con monederos electrónicos. Con esta nueva propuesta de monedero, en

la fase de pago, no es necesaria la participación de un dispositivo con módulos SAM. Esta propuesta a su vez podría ser combinada con SPEED o cualquier otro protocolo de intercambio equitativo para ofrecer una solución de pago B2C basada en monedero electrónico que solucione todos los problemas anteriores. En el caso de utilizar otro protocolo distinto a SPEED habría que tener en cuenta que los protocolos de intercambio equitativo, hasta donde nosotros sabemos, no incorporan la posibilidad de negociación. En este caso sería necesario además combinarlos con un protocolo de negociación [250].

A continuación, presentaremos cada una de estas propuestas por separado. Por cada una de estas propuestas realizaremos una descripción del sistema y un análisis de las distintas características de seguridad aportadas.

3.2. SPEED

En esta sección presentamos un nuevo protocolo de pagos para la compra de productos electrónicos denominado SPEED [249] (Smartcard-based Payment with Encrypted Electronic Delivery – pago basado en tarjeta inteligente con entrega electrónica cifrada). Este protocolo está basado en los siguientes pilares:

- Protocolo que abarca las distintas fases relacionadas con la adquisición de un producto: negociación, compra y distribución segura.
- Sistema de pago basado en el uso de tarjeta inteligente. De esta forma se facilita la movilidad del usuario (y de su dinero electrónico), se protege de forma segura el dinero electrónico así como el material criptográfico asociado, se facilita que pueda ser utilizado en distintos escenarios (compras en línea, compras en máquinas de vending o puntos de venta, etc.).
- Garantía de no repudio de las transacciones.
- Intercambio equitativo.
- Atomicidad de la transacción.
- Independencia del vendedor para el soporte de pagos en tarjeta. Se evita el rechazo de los vendedores, ya que se evita la integración de módulos hardware (o nuevos terminales) en el sistema de comercio electrónico del vendedor.

Además, para evaluar sus características, se desarrolló una implementación que permitirá la integración del protocolo en un sistema de comercio.

3.2.1. Visión general

El protocolo SPEED ha sido diseñado con el fin de facilitar la compra y la entrega de productos digitales, tales como ficheros musicales (MP3s, WMV, etc.), documentos electrónicos (revistas, artículos científicos, artículos en revistas académicas, etc.), imágenes, etc. Es decir, cualquier producto digital que sea susceptible de ser representado como un archivo y que pueda ser transmitido como un todo. Por tanto, este protocolo no sería adecuado para la distribución de productos que se proporcionan por medio de

streaming. En esta situación, el enfoque que se seguiría para la adquisición de este tipo de productos sería el que se expone a continuación. Este tipo de productos que acabamos de mencionar podría ser considerado como un servicio. En este caso, el protocolo sería utilizado para realizar la compra de las claves de seguridad que nos permitan acceder de forma segura a ese contenido o servicio. El contenido (el stream o flujo multimedia) será entregado por otro medio distinto de SPEED de forma cifrada. Si el usuario ha obtenido (pagado) las claves mediante SPEED, puede acceder a este contenido distribuido bajo streaming. Así, bajo estas circunstancias, podríamos decir que el protocolo SPEED también es de utilidad para la compra de servicios protegidos por clave.

En el diseño del sistema se ha considerado como principal objetivo que el protocolo permita el pago de los productos digitales mediante el uso de una tarjeta inteligente dotada de un monedero electrónico (WG10, CEPS o EMV). Bajo estas condiciones, el modelo propuesto estaría basado, principalmente, en prepago (WG10, CEPS), aunque también podría ser de crédito/débito (EMV). Sin embargo, se han contemplado otras posibilidades y el protocolo permitiría el pago mediante una cuenta bancaria asociada a una determinada entidad o por medio de una tarjeta de crédito/débito tradicional. De esta manera, se facilita que el vendedor pueda ofrecer distintos mecanismos de pago a sus clientes aunque, en estos escenarios, la principal opción debería ser el pago mediante el monedero, debido a las características que ofrecen las tarjetas y que fueron comentadas en la Sección 2.2.3.

Una transacción de pago conforme al protocolo SPEED implica la participación de tres entidades: el cliente, el vendedor y el broker o proveedor de servicios de pago. Como requisito básico del sistema se exigirá a todas estas entidades que estén certificadas por una autoridad de certificación o proveedor de servicios de certificación. De esta forma, cada una de las entidades poseerá una clave RSA (o cualquier otro tipo de clave que permita a la misma vez realizar operaciones de firma y de cifrado, como McEliece [62] o Rabin [230]) y un certificado X.509v3 [61] de una autoridad de certificación de confianza. En nuestro modelo no se exige que todas las entidades estén certificadas por la misma autoridad de certificación. Sin embargo, antes de realizar cualquier transacción, las distintas partes deberán asegurarse que los certificados de las otras partes involucradas en la transacción han sido emitidos por una autoridad de certificación en la que confían.

El broker podría ser una entidad financiera o una compañía que tiene distintos acuerdos con entidades financieras y que representa a un tercero de confianza para. En general, los brokers deberán ser de reconocido prestigio para poder conseguir la confianza de los clientes y vendedores en el sistema propuesto. Esta entidad ha sido introducida en el sistema por dos motivos fundamentales. El primero, para que actúe como intermediaria o tercero de confianza para el cliente y el vendedor en el proceso de pago, por lo que aportará seguridad y confianza al sistema. El segundo, último y fundamental motivo es facilitar los procesos de pago basados en tarjeta inteligente que, como se mencionó anteriormente, aumentan la seguridad y facilitan la movilidad de la información de pago.

El broker poseerá una serie de módulos SAM que son los que permitirán recibir el pago del cliente y, por tanto, realizar el decremento del saldo del monedero conforme a la cantidad pagada. De esta forma se facilita a los vendedores el soporte de pagos basados en

monedero, ya que no es necesario que en sus aplicaciones de comercio electrónico integren el uso de tales módulos. En este escenario, el broker es el encargado de mantener las cuentas de los vendedores incrementando el saldo de éstas de acuerdo a los distintos pagos recibidos. El broker también es responsable de las posibles resoluciones de disputas que puedan surgir entre clientes y vendedores. En el caso de que el broker detecte problemas con un determinado cliente o vendedor de manera habitual, tomará determinadas medidas, como por ejemplo, comunicar con la entidad financiera del cliente para revocar su monedero, no aceptar transacciones del vendedor, etc. La reputación del broker vendrá dada en función de cómo se realice este proceso de resolución de disputas y, como hemos mencionado, este aspecto es fundamental de cara establecer la confianza de los usuarios en el sistema.

El vendedor, por su parte, es la entidad que ofrece sus productos digitales a través de la red y que serán pagados con el protocolo aquí descrito. Durante el proceso de pago el vendedor facilitará, de forma segura, su número de cuenta en el broker (o un número de cuenta de otra entidad que haya sido acordado con el broker), de forma que el importe de la transacción le sea ingresado en dicha cuenta. Por lo general, en nuestro modelo, entre el vendedor y el broker existirá una relación de confianza a largo plazo de la misma forma que podría existir con un banco o emisor de tarjetas de crédito.

Finalmente, el usuario realizará la adquisición de los productos del vendedor mediante el pago al broker. El broker debe ser una entidad de confianza para el cliente, ya que será el encargado de manejar la información de pago del cliente. En general, el usuario poseerá una tarjeta inteligente con monedero electrónico para realizar pagos. Esta tarjeta podría haber sido proporcionada por una entidad financiera, un broker o una compañía privada. Sin embargo, como se mencionaba anteriormente, también podría efectuar el pago a través de una cuenta que éste posea en el broker (en este caso será necesario un registro previo del usuario con el broker) o mediante el uso de su tarjeta de crédito/débito.

El diseño del protocolo está basado en diversos estándares. Así, para la especificación de los mensajes del protocolo nos hemos basado en ASN.1 [136], como formato criptográfico para el intercambio de los mensajes elegimos PKCS#7 [245]/CMS [136] y, finalmente, para identificar a los usuarios en el sistema nos decidimos por los certificados X.509v3 [61]. En cuanto al monedero electrónico resulta interesante comentar que la propuesta es independiente del monedero elegido y podría ser utilizada tanto con WG10 [44], CEPS [45] y EMV [89, 90, 91, 92].

3.2.2. Modelo de compra

En cualquier transacción llevada a cabo con este protocolo participan las tres entidades antes mencionadas: cliente, vendedor y broker. Durante la transacción, el vendedor entrega un producto electrónico al cliente. La entrega lleva asociada el decremento del monedero electrónico (o de una cuenta asociada o no a una tarjeta crédito/débito) del cliente y el ingreso en la cuenta del vendedor del precio asociado al producto. Cualquier transacción en SPEED se puede descomponer en una serie de fases. En primer lugar se ha

definido una fase opcional de negociación del precio del producto, a continuación se produce la entrega del producto y, finalmente, el pago.

Para llevar a cabo estas transacciones de pago se han definido dos modos de operación: un *modo normal* y un modo *rápido o agresivo*. El *modo de funcionamiento normal* es el que más características ofrece, tanto desde el punto de vista funcional como desde el punto de vista de la seguridad. Desde el punto de vista funcional, este modo incluye la capacidad de negociación del precio del producto. Por otra parte, desde el punto de vista de la seguridad, este modo es capaz de evitar de una forma más eficaz ataques de denegación de servicio. El *modo de operación agresivo*, en cambio, reduce tanto el nivel de seguridad proporcionado como el número de mensajes a intercambiar, ya que está pensado para la venta de bienes cuyo tamaño es pequeño (de forma que sea más difícil producir un ataque de denegación de servicio) o para aquellos escenarios con menores requisitos de seguridad.

Como acabamos de mencionar, el proceso de compra en SPEED se puede dividir en tres fases: negociación, entrega del producto y pago. En la Figura 3.1 se muestra el intercambio de mensajes que componen la secuencia de compra en SPEED en el modo de operación normal. En este proceso, los tres primeros mensajes que intercambian cliente y vendedor constituyen la fase de negociación (solicitud de negociación, paso de negociación y acuerdo). El mensaje 4 (envío de producto cifrado) representa la fase de entrega del producto. El producto se envía cifrado con una clave simétrica generada aleatoriamente por el vendedor y que no será proporcionada al cliente hasta que éste haya efectuado el pago. Finalmente, la fase de pago la constituyen los mensajes 5 y 6 (orden de pago y recibo de compra). En esta fase de pago, si el pago se realiza por medio del monedero, el broker y el cliente intercambian una serie de mensajes adicionales destinados a llevar a cabo el proceso de decremento del monedero. Este intercambio aparece reflejado en la Figura 3.1 mediante la línea punteada entre el broker y el cliente.

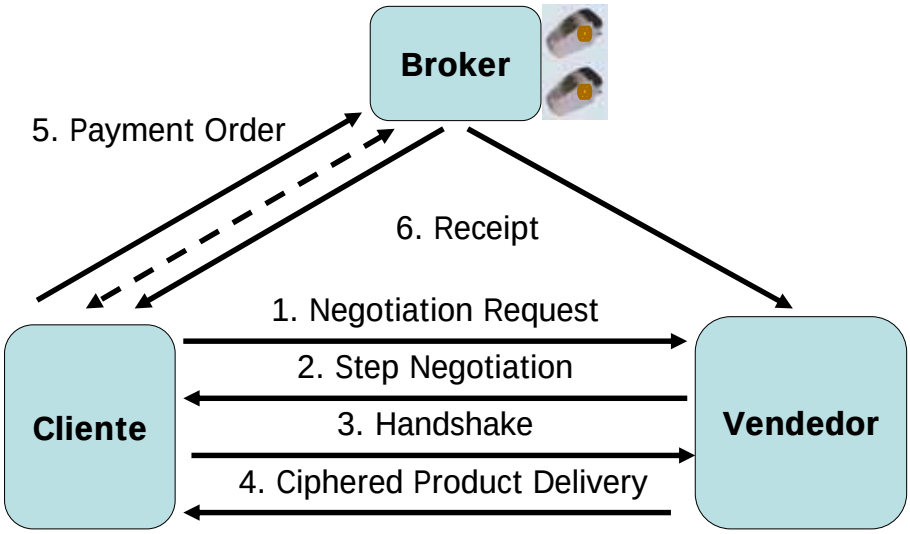


Figura 3.1. Modelo de pago en SPEED.

3.2.3. Especificación del protocolo

A lo largo de esta sección describiremos los distintos mensajes que forman parte del protocolo SPEED. El protocolo, como se mencionó anteriormente, soporta dos modos de operación. Sin embargo, antes de entrar en la descripción de los mensajes de cada modo proporcionaremos la notación que utilizaremos para expresar los intercambios de mensajes criptográficos. Una vez introducida la notación, describiremos los mensajes que forman parte de cada uno de los dos modos de funcionamiento del protocolo, es decir, el modo de funcionamiento normal y el modo rápido o agresivo.

Notación

En esta sección introducimos una notación para expresar los distintos mensajes criptográficos de las propuestas que presentaremos a lo largo de este capítulo. La mayoría de los elementos que forman parte de esta notación se utilizan habitualmente para expresar muchos protocolos relacionados con el intercambio de información de forma segura, especialmente los protocolos que garantizan el no repudio y el intercambio equitativo, como por ejemplo en [163, 183, 209, 305].

Notación	Significado
[Datos]	Indica que los *Datos* que forman parte de este mensaje son opcionales, y por tanto, podrían no estar.
H(Datos)	El resumen digital o hash de *Datos*, obtenidos por medio de un algoritmo de resumen digital resistente a colisiones como SHA2.
H_k(Datos)	Los *Datos* son autenticados usando un algoritmo HMAC con la clave simétrica *k*. Este elemento representa tanto los *Datos* como los datos de autenticación generados con la función HMAC.
E_k(Datos)	El cifrado de los *Datos* utilizando un algoritmo de cifrado simétrico como IDEA o AES con la clave *k*.
E_X(Datos)	El cifrado de los *Datos* con una función de cifrado basada en criptografía asimétrica (o pública) y con la clave pública de la entidad *X*.
S_X(Datos)	La firma de los *Datos* con una función de criptografía asimétrica utilizando la clave privada de la entidad *X*. Junto con la firma también se envían los *Datos*.
$Cert_X$	El certificado de la entidad *X*.
$PublicKey_X$	La clave pública de la entidad *X*.
$X \rightarrow Y$	Indica que *X* envía un mensaje a *Y*.

Tabla 3.1. Notación.

Modo Normal

El modo de operación normal es, en general, adecuado para los siguientes escenarios:

- El cliente no conoce el precio del producto porque no está fijado.
- El vendedor ofrece la posibilidad de negociar el precio del producto y el cliente quiere negociar un precio inferior.
- El tamaño del producto, en bytes, es grande (este tamaño variará dependiendo de la carga que pueda soportar el servidor del vendedor). En este escenario, sería posible llevar a cabo, de forma sencilla, ataques de denegación de servicio si se utilizara el modo rápido, tal y como explicaremos posteriormente en las siguientes secciones. Con el fin de evitar este tipo de ataques, este modo sería el más adecuado para productos como canciones, vídeos, etc.

A continuación detallaremos los distintos mensajes que forman parte de cada una de las fases en este modo. Estos mensajes son los que aparecen en Figura 3.1.

Fase de Negociación

Esta fase, a la misma vez que inicia el protocolo y establece la información criptográfica que será utilizada en los distintos mensajes, permite la negociación del precio del producto. Está compuesta de tres mensajes: *NegotiationRequest (solicitud de negociación)*, *NegotiationStep (paso de negociación)* y *Handshake (acuerdo)*.

Los dos primeros mensajes permiten realizar la negociación del precio del producto. En general, la negociación seguirá un patrón oferta-contraoferta (petición-respuesta). En el primer mensaje (*NegotiationRequest*), el cliente realiza una propuesta de precio por el producto. A continuación, el vendedor responde con su propuesta (*NegotiationStep*). A partir de este momento, la negociación puede continuar hasta que el cliente y el vendedor acuerden el precio o una de las partes decida romper la negociación. Para los sucesivos pasos de negociación, tanto el cliente como el vendedor utilizarán el mensaje *NegotiationStep*. Finalmente, el cliente envía el mensaje *Handshake* cuando haya decidido aceptar el precio propuesto por el vendedor.

El cliente envía el mensaje de *NegotiationRequest* la primera vez que el cliente y el vendedor negocian por un determinado producto. Este mensaje está firmado digitalmente con la clave privada del cliente y cifrado con la clave pública del vendedor. Tal y como argumentan Abadi y Needham en [1] y Stallings en [282] para mensajes que están firmados y cifrados es mejor realizar las operaciones en el orden que acabamos de mencionar.

1. C → V: NegotiationRequest (Solicitud de Negociación)

$$E_V(S_C(C_{ID}, NID, SeqN, ProductID, [Price], V_{ID}, enkey, signkey, Flag, [Credentials]))$$

donde:

- *C (Cliente)*: Representa al cliente.
- *V (Vendedor)*: Representa al vendedor.
- C_{ID} (*Identificador del Cliente*). Este campo representa el resumen digital de la clave pública del cliente. Identifica de forma unívoca el cliente que envía esta solicitud de negociación. Este identificador se incluye siguiendo los principios de diseño propuestos por Abadi y Needham [1], y Gürgens et al. [120] para protocolos criptográficos. De esta forma podemos evitar diversos ataques tales como los que se mencionan en las referencias que acabamos de citar. Estos ataques también serán comentados en la Sección 3.2.4.
- *NID (Identificador de Negociación)*. Es un valor de al menos 4 bytes generado por el cliente por medio de un generador de números pseudo-aleatorio. Para su generación se deberían seguir los criterios expuestos en [79, 164]. Este campo identifica de manera unívoca la negociación que se está llevando a cabo entre el cliente y el vendedor. Aunque este identificador no es globalmente único (podría ser utilizado con otros vendedores), se utiliza para distinguir entre las distintas negociaciones que en un momento dado pueden estar llevándose a cabo con el mismo vendedor.
- *SeqN (Número de secuencia)*. Durante la fase de negociación, es posible el intercambio de varios mensajes. Cada mensaje en esta secuencia debe ser único con el fin de evitar ataques de reenvío. Por esta razón, este campo está presente en todos los mensajes de negociación y cada participante debe incrementar su valor después de recibir este tipo de mensaje.
- *ProductID (Identificador de Producto)*. Es una cadena de texto compuesta de un código de producto y una descripción del producto conforme a la realizada por el vendedor.
- *Price (Precio)*. Es el precio que el cliente está dispuesto a pagar por el producto. Este campo es opcional en este mensaje ya que el cliente podría no conocer el precio del producto. El precio está expresado como un valor en una determinada divisa de acuerdo al estándar ISO 4217 [141].
- V_{ID} *(Identificador del Vendedor)*. Este campo representa el resumen digital de la clave pública del vendedor. Identifica de forma unívoca al vendedor al que va dirigida esta solicitud de negociación. Este identificador se incluye siguiendo los principios de diseño propuestos por Abadi y Needham [1], y Gürgens et al. [120]. Así, protegemos al cliente de posibles ataques de suplantación que podrían llevar a cabo vendedores mal intencionados (este ataque se explica en más detalle en la Sección 2.3.4).
- *enkey (Clave de Cifrado)*. Es una clave simétrica que se emplea para proporcionar confidencialidad en el intercambio de mensajes que se producirá en los siguientes mensajes de negociación y acuerdo. La clave se genera conforme al algoritmo de cifrado simétrico y a la longitud que se considere más adecuada. Para el cifrado con esta clave se podrían utilizar algoritmos tales como IDEA, AES, RC6, etc.

- *signkey (clave de Firma)*. Se trata de una clave simétrica que tiene como fin proveer de integridad a los mensajes que posteriormente intercambiarán cliente y vendedor durante la negociación y acuerdo. Esta clave, que genera el cliente de forma aleatoria, será del mismo tipo de clave que la clave de cifrado. Como función para calcular la integridad de los mensajes utilizaremos la función HMAC con el algoritmo de resumen digital que se considere más adecuado, como podría ser SHA-2.
- *Flag (Indicador de Fin de Negociación)*. Este campo contiene un valor booleano que indica si el precio propuesto representa la última oferta del cliente. De esta forma, permitimos indicar si el cliente está dispuesto (el flag con el valor *true*) o no (el flag con el valor a *false*) a negociar otros precios.
- *Credentials (Credenciales)*. La estrategia de negociación puede ser modificada mediante la presentación de credenciales. En el caso que se utilicen, éstas pueden ir incluidas en este campo y tanto su uso como su formato son totalmente independientes al protocolo.

Algunos de estos campos aparecerán nuevamente en el documento. Sin embargo, salvo que su semántica cambie, no volverán a ser explicados.

A continuación, tanto el vendedor como el cliente utilizarán el siguiente mensaje, *NegotiationStep*, para acordar el precio final del producto. La información que se intercambia en este mensaje está protegida por medio de criptografía simétrica, utilizando las claves proporcionadas por el cliente en el anterior mensaje. En concreto, el mensaje es autenticado utilizando la función HMAC y la clave *signkey* y, después, es cifrado utilizando el algoritmo de cifrado simétrico elegido con la clave *enkey*. Con este mensaje, cada participante realiza una oferta que, a lo largo de los distintos pasos de negociación, podría variar dependiendo de la estrategia de negociación escogida.

2. V → C: NegotiationStep (Paso de Negociación)

$$E_{enkey}(H_{signkey}(V_{ID}, C_{ID}, NID, SeqN, Price, Flag))$$

donde:

- *Price (Precio)*. En este caso indica el precio propuesto por el vendedor o por el cliente.
- *Flag (Indicador de Fin de Negociación)*. Representa, mediante un valor booleano, si es la última oferta de la entidad que está enviando este mensaje.

Este mensaje tiene por objetivo permitir negociar a ambas partes, de manera rápida, el precio final del producto, de tal forma que el cliente intentará obtener el precio más bajo, mientras que el vendedor intentará obtener el precio más alto posible. Este mensaje será intercambiado en la negociación hasta que se cumpla alguna de las condiciones siguientes:

- El cliente acepta el precio propuesto por el vendedor en el último mensaje *NegotiationStep*.
- El cliente o el vendedor reciben un mensaje de *NegotiationStep* que contiene el indicador de fin negociación activado y éste no está de acuerdo con la oferta. En este caso se da por rota la negociación y, por tanto, la comunicación finaliza con el mensaje de *Abortar Negociación*. Este mensaje también se utiliza cuando una de las partes después de enviar un mensaje, no recibe una respuesta válida antes de un periodo de tiempo razonable. Este mensaje tiene el siguiente formato:

$$X \rightarrow Y: \text{StopNegotiation (Abortar Negociación)}$$

$$E_{enkey}(S_X(X_{ID}, Y_{ID}, NID, Abort, [Reason]))$$

donde:

  - *Abort (Abortar)*. Flag que indica que la negociación se aborta.
  - *Reason (razón)*: Campo opcional que se puede utilizar para indicar por la razón por la que se aborta el protocolo.

Una vez que el cliente y el vendedor han acordado el precio del producto, el cliente envía el mensaje *Handshake* al vendedor para notificar que el cliente acepta su última oferta. A este mensaje se podría llegar después de varios pasos de negociación o podría ser la respuesta a la primera oferta realizada por el vendedor si el cliente está de acuerdo con la oferta propuesta.

El mensaje *Handshake* está firmado por el cliente usando su clave privada. Así el vendedor no puede generar este mensaje en lugar del cliente por medio de las claves simétricas. Además, el mensaje se envía cifrado usando la clave *enkey*.

$$3.\ C \rightarrow V: \text{Handshake (Acuerdo)}$$

$$E_{enkey}(S_C(C_{ID}, V_{ID}, NID, Price))$$

Con este mensaje se daría por concluida la fase de negociación ya que las partes han llegado a un acuerdo por el precio final del producto y, además, el cliente confirma que está dispuesto a pagar esa cantidad por ese producto.

Fase de Entrega

Esta fase está compuesta de un único mensaje denominado *CipheredProductDelivery (Envío del Producto Cifrado)*. Este mensaje lo envía el vendedor una vez que ha recibido el mensaje *Handshake* por parte del cliente. En ese caso, el vendedor construye un mensaje que contiene el producto solicitado por el cliente cifrado con una clave simétrica *k*

que fue generada de forma aleatoria. Junto con el producto cifrado, el vendedor adjunta una factura que está digitalmente firmada por él y que contiene toda la información para identificar de forma unívoca la transacción. Además, la factura contiene la información necesaria para que el broker, posteriormente, le pueda ingresar la cantidad que el cliente pagó por el producto. Esta factura se envía cifrada al cliente con la clave *enkey*. Estos elementos, que forman parte de este mensaje, se muestran continuación.

4. V → C: CipheredProductDelivery (Envío del Producto Cifrado)

$E_k(Product)$, $E_{enkey}(S_V(V_{ID}, C_{ID}, Bill))$

donde:

- *$E_K(Product)$*. Representa el producto cifrado que solicitó el cliente. La clave *k* es una clave generada aleatoriamente de acuerdo al algoritmo de cifrado simétrico que se considere más adecuado. La clave se genera en el momento en que se alcanza el acuerdo y se inicia la construcción de este mensaje. Cuando el pago se ha realizado, el cliente recibirá la clave para descifrar el producto.
- *Bill (Factura)*. La factura está compuesta de los siguientes campos que se muestran a continuación:

  $Bill = E_B(AccountNumber, PaymentOrderID, k), H(E_k(Product)),$
  $H(ProductID), PaymentOrderID, DateTime, Price, reckey$

  donde:

  - B (Broker). *Representa al Broker.*
  - *AccountNumber (Número de Cuenta)*. Es el número de cuenta del vendedor donde se ingresa el importe del pago. Este número podría corresponder con el número que el vendedor utiliza en una determinada entidad bancaria o podría ser un número simbólico que le haya asignado el broker y que se corresponda con un número de cuenta que tendrá almacenado éste. El vendedor podría utilizar distintos números de cuenta con el broker de forma que tuviera separados los ingresos según su propio interés (por tipo de producto, por sucursal, etc.).
  - *k*. Clave simétrica para descifrar el producto. Tanto esta clave como el número de cuenta se envían cifrados con la clave pública del broker. De esta forma, por un lado se mantiene la confidencialidad de la información bancaria del vendedor y, por otro lado, se evita que el cliente pueda conocer la clave con la que se cifró el producto hasta que no realice el pago.
  - *$H(E_k(Product))$* es el resumen digital del producto cifrado de acuerdo con el algoritmo de resumen digital que se considere oportuno.

- *H(ProductID)* representa el resumen digital del campo *ProductID* incluido en el mensaje de *NegotiationRequest*. Se utiliza el resumen digital en vez del valor del campo con el fin de ocultar la descripción del producto al broker y así mantener la privacidad del usuario. Posteriormente, si el producto que recibió el cliente no coincidiese con la descripción especificada, el cliente podría utilizar este campo para reclamar el producto que originalmente solicitó.
- *PaymentOrderID (Identificador de la Transacción de Pago)* es un identificador de la transacción actual entre el cliente y el vendedor. Este identificador debe ser único en el sistema, teniendo en cuenta todas las transacciones de todos los vendedores. Está compuesto de varios elementos. El primero es el campo NID asociado a esta transacción entre el cliente y el vendedor. A continuación, vienen los identificadores del cliente y del vendedor y, finalmente, el resumen digital del producto cifrado (siguiendo las recomendaciones propuestas en [120]). Por lo que, PaymentOrderID = NID, ID_C, ID_V, $H(E_k(Product))$.
- *DateTime (Fecha y hora)*. Contiene la fecha y la hora en la que se emite la factura.
- *reckey (clave de recepción)*. Es una clave simétrica del mismo tipo que las claves *enckey* y *signkey* y que se genera aleatoriamente por el vendedor. Como posteriormente veremos, esta clave se utiliza para enviar de forma cifrada el recibo de la transacción a cliente y vendedor.

Una vez que el cliente ha recibido este mensaje se da por finalizada la fase de entrega del producto y, en ese momento, se inicia la fase de pago.

Fase de Pago

El número de mensajes de que consta esta fase es variable dependiendo del método de pago elegido (pago con monedero, cuenta o tarjeta de crédito/débito). Aún así, independientemente del modo seleccionado, hay dos mensajes que son comunes a todos ellos: el mensaje *PaymentOrder (Orden de Pago)* y el mensaje *PurchaseReceipt (Recibo de Compra)*.

El mensaje *PaymentOrder* lo envía el cliente al broker para iniciar la fase de pago. El pago se podría realizar con sólo este mensaje, o podrían requerirse mensajes adicionales dependiendo del modo de pago.

Una vez que el pago se ha efectuado, con el mensaje *PurchaseReceipt*, el broker envía un recibo de compra que incluye la clave simétrica utilizada para cifrar el producto, así como determinada información referente a la transacción que acaba de producirse y que podría ser utilizada para resolver disputas. Con este mensaje finalizaría la fase de pago y, además, el proceso de compra por medio del protocolo SPEED.

La información del mensaje *PaymentOrder* es la que se muestra a continuación.

5. C → B: PaymentOrder (Orden de Pago)

$E_B(S_C(C_{ID}, B_{ID}, S_V(V_{ID}, C_{ID}, Bill), PaymentMode))$

donde:

- *B_{ID} (Identificador del Broker)*. Identificador con las mismas características que C_{ID} y V_{ID}.
- *Bill (Factura)*. El cliente incluye la factura previamente recibida por el vendedor en el mensaje *CipheredProductDelivery*.
- *PaymentMode (Modo de Pago)*. La estructura de este campo depende del método de pago elegido por el cliente. Las opciones disponibles son:
  - {AccountNumber}. Si el cliente elige pagar por medio de una cuenta que el cliente tenga con el broker y otra entidad financiera. En este caso el campo *AccountNumber* indica el número de la cuenta.
  - {CreditDebitNumber, Name, ExpDate, [CVC]}. Otra opción es permitir que el usuario realice el pago por medio de su tarjeta de crédito/débito. En este caso, los distintos campos contienen la información que se necesita para realizar este tipo de pago. Es decir, el número de la tarjeta de crédito/débito (*CreditDebitNumber*), el nombre del usuario tal y como aparece en su tarjeta (*Name*), la fecha de caducidad asociada a la tarjeta (*ExpDate*) y, finalmente, y de forma opcional, el código de verificación del cliente (*CVC*).
  - {benckey, bsignkey, [SCmessage]}. Esta es la información que se envía en el caso de que el pago se realice con el monedero. Como se comentó la Sección 2.2.4), el pago con el monedero, en general, supone el intercambio de una serie de mensajes entre éste y un módulo SAM que, en este caso, posee el broker. Estos mensajes forman también parte del protocolo SPEED y, por tanto, la información que contienen será protegida igual que el resto de los mensajes de este protocolo. Para este propósito, en el mensaje *PaymentOrder* se proveen dos claves simétricas que serán utilizadas para proteger el resto de los mensajes a intercambiar con la finalidad de realizar el decremento del saldo del monedero. Una servirá para autenticar los mensajes (bsignkey) y la otra se utilizará para cifrarlos (benckey). Junto con estas dos claves, en la orden de pago también se envía el primer APDU (*Application Protocol Data Unit* – Unidad de Datos del Protocolo de Aplicación o mensajes de tarjeta) (o conjunto de APDUs) a intercambiar entre el monedero y el módulo SAM, es decir, el APDU *SCmessage*. En general, este APDU se utiliza para autenticar el monedero frente al módulo SAM.
  - {e-coin}. En este caso el cliente utiliza una moneda electrónica para realizar el pago al broker. Esta moneda podría proceder de alguno de los esquemas introducidos en la Sección 2.2.2, como eCash, Paycash, etc.

Básicamente, el campo e-coin (moneda electrónica) representa una secuencia de bytes que codifica una moneda de acuerdo a los esquemas que acabamos de mencionar o con cualquier futuro esquema que pueda aparecer.

Cuando el método de pago elegido es el monedero, es necesario intercambiar al menos dos mensajes adicionales para realizar el pago. El primero de esos mensajes es el que permite que el módulo SAM envíe un APDU a la tarjeta inteligente. A este primer mensaje se le denomina *SAMModuleMessage (Mensaje del Módulo SAM)* y contiene un APDU que llamaremos *SAMmessage* y que, en general, permite realizar la autenticación del módulo SAM frente a la tarjeta. Este APDU lo genera el módulo SAM en respuesta al APDU contenida en *SCmessage*.

a. B → C: SAMModuleMessage (Mensaje del Módulo SAM)

$$E_{benckey}(H_{bsignkey}(B_{ID}, C_{ID}, \text{PaymentOrderID}, \text{SAMmessage}))$$

A continuación, el cliente envía el mensaje *SAMmessage* al monedero. En general, en este paso, en caso de que la autenticación sea satisfactoria, se procedería a realizar el decremento del monedero y se generaría un mensaje que confirmase dicho decremento. Este nuevo mensaje de la tarjeta iría dentro del campo *SCmessage* que definimos anteriormente. A continuación, el cliente crea un mensaje que contiene a *SCmessage* como uno de sus campos, este nuevo mensaje se llamará *PurseMessage (Mensaje del Monedero)*. En general, este paso dará por concluido el pago con el monedero.

En caso de que para un monedero específico fueran necesarios más pasos de intercambio, se utilizarían los mensajes *SAMModuleMessage* y *PurseMessage* tantas veces como fuera necesario hasta completar el decremento de dicho monedero.

b. C → B: PurseMessage (Mensaje del Monedero)

$$E_{benckey}(H_{bsignkey}(C_{ID}, B_{ID}, \text{PaymentOrderID}, \text{SCmessage}))$$

Una vez que el decremento del monedero se ha realizado correctamente, el broker procede a enviar al cliente y al vendedor el recibo de la transacción efectuada. El recibo contiene el identificador de la transacción, el importe pagado y, lo más importante para el cliente, la clave *k* que le permitirá descifrar el producto por el que pagó. Este recibo se envía firmado por el broker y cifrado con la clave simétrica *reckey* que estaba contenida en la factura y que conocen tanto cliente como vendedor.

6. B → C, V: PurchaseReceipt (Recibo de compra)

$$E_{reckey}(S_B(B_{ID}, C_{ID}, V_{ID}, PaymentOrderID, Price, k))$$

Una vez recibido este mensaje, el cliente obtendrá la clave *k* y pasará a descifrar el producto cifrado. Si la clave no permite descifrarlo o si el producto que recibió no se corresponde con la descripción, utilizará el recibo junto con la descripción del producto para presentar una queja.

Modo Rápido o Agresivo

En este modo no hay fase de negociación ya que el cliente directamente acepta el precio propuesto por el vendedor. En este caso, el cliente recibe el producto cifrado y la factura de manera inmediata.

Este modo de funcionamiento es adecuado para aquellos casos en el que la negociación no tiene sentido, por ejemplo, porque el precio está fijado, o en aquellos casos donde el número de transacciones por segundo es lo más importante. Este modo también se podría utilizar en aquellos en los que se considere que éste se puede utilizar para producir ataques de denegación de servicio, por ejemplo, solicitando productos de un gran tamaño por los que posteriormente no se realizará un pago.

El modo rápido reduce el intercambio de mensajes de seis a cuatro (ver Figura 3.2), de tal manera que ya no son necesarios los mensajes *NegotiationStep* ni *Handshake*. Para soportar este modo ha sido necesario, además, modificar los mensajes *NegotiationRequest* y *CipheredProductDelivery*. A continuación, se muestran los cambios que se han realizado en cada uno de ellos. Adicionalmente, como la semántica de la primera fase ha cambiado también ha cambiado su nombre. En este modo, a la primera fase se la denomina *Solicitud de Producto*. En el resto de los mensajes no ha sido necesario realizar modificaciones adicionales.

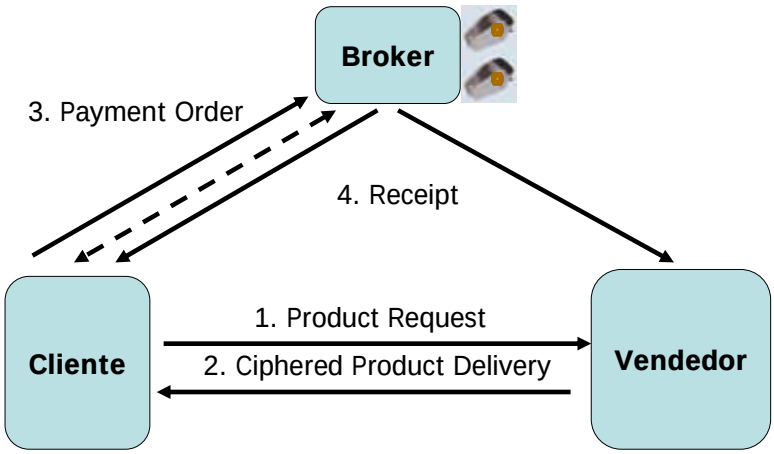


Figura 3.2. Modo agresivo de compra en SPEED.

Fase de Solicitud de Producto

En esta fase sólo se ha definido un mensaje denominado *ProductRequest (Solicitud de Producto)* y que, como se comentaba anteriormente, es una modificación del mensaje *NegotiationRequest*. La principal diferencia entre un mensaje y otro radica en los campos relacionados con la información de negociación: una de las claves de negociación (*signkey*), el número de secuencia (*SeqN*) y el indicador de fin de negociación (*Flag*).

1. C → V: ProductRequest (Solicitud de Producto)

$$E_V(S_C(C_{ID}, V_{ID}, NID, ProductID, Price, enkey))$$

Una vez que el vendedor recibe este mensaje se pasa a la fase de entrega del producto.

Fase de Entrega del Producto

Debido a que en el anterior mensaje se eliminaron las claves de negociación, ahora la diferencia entre el mensaje *CipheredProductDelivery* del modo normal y el modo rápido es que la factura firmada por el vendedor en el modo rápido se envía cifrada con la clave pública del cliente. Sin embargo, el contenido sigue siendo el mismo.

2. V → C: Envío del Producto Cifrado

$$E_k(Product),\ E_{enkey}(S_V(V_{ID}, C_{ID}, Bill))$$

3.2.4. Análisis de las propiedades del protocolo

En esta sección analizamos las distintas propiedades que satisface el protocolo SPEED. En primer lugar será analizado desde el punto de vista de la seguridad, mostrando cuáles son los posibles ataques que se podrían producir y cómo el protocolo ofrece medidas de seguridad ante ellos. Además, este análisis tiene en cuenta diversos aspectos como son la visibilidad, los posibles fraudes que podrían intentar cometer tanto el cliente como el vendedor, y cómo se daría respuesta a las diversas reclamaciones que pueden presentarse en el sistema. Una vez analizada una de las propiedades más importantes que debe cumplir un protocolo, es decir, la seguridad, pasaremos a analizar otro tipo de propiedades que son importantes y que fueron mencionadas en la Sección 2.2.1.

Suposiciones sobre la Criptografía

Previo a analizar los distintos aspectos de seguridad las suposiciones realizadas con respecto a las funciones criptográficas que se utilizan en el protocolo SPEED.

- **Cifrado opaco**. El cifrado se supone que es opaco, es decir, si un mensaje ha sido cifrado con la clave pública de la entidad X, sólo ésta puede obtener el mensaje.
- **Firmas infalsificables**. Supondremos que una firma no se puede falsificar. De forma que, si un mensaje ha sido firmado por una determinado entidad, cualquiera que

tenga la clave pública asociada a ese participante podrá verificar que el mensaje fue, en efecto, firmado por tal entidad.

- **Funciones de resumen libres de colisiones**. Supondremos que las funciones de resumen están libres de colisiones.
- **Autoridad de certificación (CA – Certification Authority) de confianza**. En el modelo suponemos que existe al menos una autoridad de certificación en la que todas las partes pueden confiar. El certificado de esta entidad estará disponible para las distintas partes y, además, permitirá verificar los certificados emitidos por esta entidad.

Validación formal

Para verificar la seguridad del protocolo hemos realizado una validación formal de nuestro protocolo basada en el uso de la herramienta AVISPA (Automated Validation of Internet Security Protocols and Applications) [9, 11, 303]. Esta herramienta nos permite validar formalmente algunas de las propiedades que presentaremos en los siguientes apartados.

A continuación proporcionamos una descripción del proceso seguido. En primer lugar, especificamos el protocolo en el lenguaje HLPSL (High Level Protocol Specification Language – Lenguaje de Especificación de Protocolos de Alto Nivel) [53]. A continuación, la herramienta AVISPA lo transforma de acuerdo a un formato intermedio, llamado IF (Intermediate Format Specification) [53]. Finalmente, la especificación IF se analiza invocando varios módulos que la herramienta proporciona que nos permiten comprobar la especificación IF contra distintos tipos de ataques conocidos.

Los módulos proporcionados en AVISPA son: un analizador de modelos simbólicos, On-the-fly Model Checker (OFMC) [9, 16]; un resolutor de restricciones, Constraint-Logic-based Attack Searcher (CL-AtSe) [9, 294]; un analizador de modelos basado en una reducción de los problemas de inseguridad de un protocolo a problemas de satisfacibilidad en lógica proposicional (SAT) [8], SAT-based Model Checker (SATMC) [7]; y una herramienta de análisis de protocolos basada en autómatas de árboles, Tree Automata-based Protocol Analyzer (TA4SP) [28].

Los módulos mencionados analizan el protocolo bajo la suposición de criptografía perfecta y que los mensajes del protocolo se intercambian sobre una red de acuerdo al modelo de intruso (o atacante) Dolev-Yao [74]. Es decir, estos módulos analizan el protocolo considerando el modelo estándar asíncrono e independiente del protocolo de un intruso que controla la red y que no puede romper la criptografía [303]. En concreto, el intruso o atacante puede interceptar los mensajes, analizarlos si posee las claves correspondientes para su descifrado y generar mensajes a partir del conocimiento que adquiere en los intercambios de mensajes y enviarlos como si fuera otra parte o entidad. Con esta herramienta podemos utilizar técnicas de análisis automático, tales como falsificación de protocolos o verificación basada en abstracción. En concreto, AVISPA nos permite determinar si la máquina de estados del protocolo está correctamente diseñada

(protocolos no determinísticos), ataques de reenvío, confidencialidad, ataques de suplantación, mantenimiento de información secreta y autenticación (débil y fuerte).

Como salida al análisis del protocolo con un módulo, la herramienta produce el resultado de su análisis en un formato de salida preciso, indicando si se han satisfecho o no los requisitos de seguridad establecidos para el protocolo. En caso de que se haya encontrado un posible ataque la herramienta muestra una traza del mismo.

Siguiendo este proceso, el protocolo aquí propuesto ha sido analizado con los distintos módulos que proporciona AVISPA y podemos afirmar que la herramienta no ha encontrado ningún ataque. La especificación de SPEED en HLPSL se encuentra en el Apéndice B.

Confidencialidad

Con el fin de conseguir que la información intercambiada entre todas las partes no pueda ser accedida por terceros, todos los mensajes enviados están cifrados. Nuestra propuesta está basada en el uso de criptosistemas que cumplen el principio de privacidad perfecta [276, 285]. Un sistema cumple esta propiedad si el atacante no puede obtener ninguna información sobre el mensaje a partir de observar el texto cifrado. En SPEED se hace uso tanto de cifrado asimétrico como simétrico para conseguir este fin. De hecho esta propiedad ha sido verificada con AVISPA. Siempre que es posible, las comunicaciones se cifran mediante claves simétricas para conseguir un mayor rendimiento y velocidad en la comunicación que la que podríamos obtener con clave asimétricas.

El cifrado simétrico se utiliza en las fases de negociación del precio del producto, en la entrega cifrada del producto y en el envío del recibo. Este cifrado se realiza con una clave generada aleatoriamente por el cliente (*enkey*) de acuerdo a un algoritmo de cifrado simétrico tal como IDEA, AES o RC6. En este caso la clave podría considerarse segura para este propósito. Esta clave sólo se utilizará para una única transacción con el fin de dificultar el proceso de obtención de datos que permita averiguar la clave utilizada para un determinado mensaje. Además, así se consigue que, aún en el caso de que se averigüe una clave, la información a la que pueda acceder el atacante sea limitada (una única transacción). El resto de los mensajes están cifrados por medio de envolturas digitales como un medio para optimizar el envío de información con criptografía asimétrica.

Sin embargo, no sólo es suficiente que el algoritmo utilizado para realizar el cifrado sea seguro contra la escucha de las comunicaciones para obtener los mensajes y aplicar ataques de texto plano escogido o de diccionario (conocidos como ataques pasivos). Adicionalmente, es necesario que el cifrado proporcionado sea resistente frente ataques activos como, por ejemplo, ataques de reenvío, que serán analizados posteriormente, o ataques de copiado y pegado (cut-and-paste). Para conseguir este propósito, antes de cifrar el contenido de los distintos mensajes se han aplicado métodos de autenticación tal y como se explica en la siguiente sección. Este procedimiento de firmar y luego cifrar (y no viceversa) se recomienda en [1, 282].

Autenticación

Todos los mensajes del protocolo están autenticados y además lo hemos podido comprobar con AVISPA. El modo en que se realiza esta autenticación dependerá del mensaje. Algunos mensajes están autenticados haciendo uso de criptografía simétrica por medio de una función HMAC con la clave simétrica *signkey* que fue generada por el cliente. Básicamente, esta técnica se utiliza para autenticar los mensajes de la fase negociación del precio de producto. Por otro lado, también se emplea autenticación basada en criptografía asimétrica (por medio de firma digital) y certificados para aquellos casos en los que se quiere garantizar la autenticación de una de las partes en la comunicación. Este tipo de autenticación es necesaria cuando no se ha establecido un contacto previo, o cuando se quiere garantizar el no repudio de la transacción. Por ejemplo, en el primer mensaje, *NegotiationRequest*, el cliente se autentica mediante una firma digital que hace uso de su clave privada. De esta forma, el vendedor podrá estar seguro de que el usuario con el que establece la comunicación es el usuario que ha firmado tal mensaje. El no repudio será analizado posteriormente en más detalle.

También se utiliza una firma digital cuando el vendedor envía la factura; de esta forma, cuando el broker reciba la información sobre la clave y dónde se debe ingresar el dinero de la transacción, estará seguro de que no ha sido modificada.

Ataques de reenvío

El uso de técnicas de autenticación, como pueden ser las funciones HMAC, no garantiza que un atacante pueda reenviar antiguos mensajes. A continuación, mostramos algunos escenarios que podrían proporcionar ciertos beneficios a un atacante:

- Un vendedor malintencionado (un atacante) podría estar interesado en que un determinado cliente y vendedor no llegasen a un acuerdo. Con este fin, podría capturar los mensajes intercambiados entre el cliente y el vendedor y, así, obtener la primera oferta que hizo el vendedor al cliente. Este mensaje, posteriormente, podría ser reenviado al cliente, de forma que éste vería que el vendedor realmente no está dispuesto a negociar, puesto que sigue ofreciendo los mismos precios que al principio de la negociación. Seguramente este hecho causaría la ruptura de las negociaciones y, por tanto, el posible acuerdo. Para evitar esta situación, en los mensajes de negociación se ha insertado un número de secuencia (campo *SeqN*). Este número se incrementa cada vez que se envía un mensaje. Así, si alguna de las partes recibe un mensaje con un número de secuencia repetido, será considerado un ataque activo y, por tanto, el mensaje será descartado.
- Un cliente recibe más de una vez la misma factura. Con el reenvío de este mensaje, el vendedor persigue estafar al cliente vendiéndole repetidas veces el mismo producto que se negoció. Para evitar esta situación, en la factura se ha introducido un número único de transacción (campo *PaymentOrderID*). Así, el cliente sólo tiene que guardar el registro de las transacciones en las que ya ha participado para determinar si esa factura ya fue pagada o no. Incluso, se podría simplificar más aún y podríamos decir que el cliente sólo tendría que guardar la información de las

negociaciones a las que ha llegado a un acuerdo y todavía no ha recibido la factura. Con lo cual la información a almacenar y comprobar es menor. Un beneficio similar al anterior ataque se podría obtener presentado facturas de anteriores transacciones al broker. Este reenvío se haría con el fin de intentar que se produzcan repetidos pagos por el mismo producto. En este sentido es necesario que el broker guarde un registro de todas las transacciones que han tenido lugar recientemente.

- Mediante el reenvío de mensajes también se puede conseguir un ataque de denegación del servicio al vendedor. Esta situación se daría si se reenviara repetidamente algún mensaje de *Handshake* ya que su recepción implica el envío de un producto cifrado que en algunos casos puede tener un importante tamaño. El reenvío de muchos productos de gran tamaño podría causar problemas de rendimiento en el vendedor atacado. Para evitar este ataque sólo es necesario el vendedor mantenga un registro de las entregas ya realizadas (mediante el campo *NID* de los mensajes de *Handshake*).

Por tanto, los ataques de reenvío en este protocolo pueden ser detectados bien por los números de secuencia utilizados en la negociación o bien por el identificador único de la transacción. Gracias a estos elementos AVISPA no ha encontrado ningún ataque de reenvío.

Suplantación

Un ataque de suplantación intenta convencer a alguna de las entidades de que la comunicación sólo se está realizando entre dos entidades cuando, en realidad, existe una tercera parte que también está participando en la comunicación (suplantando a alguna de las entidades mencionadas).

Para evitar los ataques de suplantación, hemos seguido los principios básicos postulados por Abadi y Needham [1] para protocolos criptográficos. Uno de estos principios está relacionado con la identificación: "si la identidad de una entidad es esencial para el significado de un mensaje, es prudente mencionar el nombre de los principales explícitamente en el mensaje". Siguiendo este principio, incluimos el identificador del vendedor (campo V_{ID}) en los mensajes de *NegotiationRequest* y *ProductRequest*. Si no incluyéramos este identificador, se podría producir la siguiente situación, siendo M un vendedor malintencionado:

1. C → M → V: NegotiationRequest (Solicitud de Negociación)

   $E_V(S_C(NID, SeqN, ProductID, Price, enkey, signkey, Flag))$

2. V → M → C: NegotiationStep (Paso de Negociación)

   $E_{enkey}(H_{signkey}(NID, SeqN, Price, Flag))$

En esta situación, el vendedor malintencionado se ha situado entre el cliente y el vendedor y ha conseguido la información acerca de las claves que le permitirán conocer toda la información intercambiada durante la negociación. Sin embargo, el cliente ignora que no ha establecido la conexión con el vendedor correcto y el vendedor honesto tampoco sabe que hay una tercera parte entre ambos (lo que se conoce como un ataque de man-in-the-middle [282]). Sin embargo, utilizando el campo V_{ID} que identifica al vendedor deseado, cuando un mensaje *NegotiationRequest* sea reenviado y no sea para el vendedor indicado en el mensaje, será considerado como un ataque. Este tipo de ataque también lo comprueba AVISPA y en SPEED no puede llevar a cabo.

Privacidad y trazabilidad

En un protocolo de pago, hay información privada que sólo deber ser visible por aquellas partes que realmente necesiten conocerla para llevar a cabo alguna determinada tarea. Por ejemplo, la cuenta del vendedor es información que el cliente no debe conocer. Asimismo, el broker no tiene por qué conocer la descripción del producto solicitado por el cliente. Para evitar este tipo de situaciones, se cifra aquella información que no debe ser conocida por una determinada entidad. Por ejemplo, en la factura la cuenta del vendedor se cifra con la clave pública del broker de tal modo que el cliente no podrá acceder a esta información.

Otra técnica que se ha utilizado para mantener la privacidad es el uso de funciones de resumen, de tal forma que, en lugar de enviar la información como tal, se envía un resumen de dicha información. De esta forma, la entidad que reciba el mensaje no podrá conocer la información, pero si tendrá un mecanismo que permitirá relacionar la información original (en caso de que sea necesario) con el resumen digital. Por ejemplo, en la factura, en vez de enviar al broker la descripción del producto, se envía un resumen de tal descripción. Así se evita que el broker pueda obtener una traza de los productos adquiridos. En este caso, sólo podrá conocer que mantiene relación con un determinado vendedor. Posteriormente, en caso de disputas, tanto el cliente como el vendedor podrían proporcionar al broker la descripción original y éste podría comprobar, mediante el resumen, que realmente se trata de ese producto.

Finalmente, la información de pago del cliente no es conocida por el vendedor. El pago tiene lugar entre el cliente y el broker, por lo que el cliente no envía ninguna información relacionada con su cuenta o su monedero al vendedor.

Fraudes

En esta sección analizamos distintos tipos de fraude que podrían intentar cometer las distintas partes. Asimismo, comentamos los mecanismos que proporciona el protocolo para evitarlos.

En primer lugar, el vendedor, durante la entrega del producto, podría cambiar la información enviada en la factura acerca del producto negociado o su precio. Para un vendedor malicioso sería particularmente interesante cambiar el producto entregado (por ejemplo, por otro de menor valor) o el precio del producto con la finalidad de obtener un mayor beneficio. Sin embargo, el cliente puede detectar inmediatamente cualquier cambio

en la descripción del producto (en realidad, en el resumen digital del producto, que es lo que se envía con la factura) o en el precio del producto, puesto que el cliente conoce tanto la descripción del producto (y por tanto puede calcular el resumen digital) como el precio del producto ya que fue acordado con el mensaje *Handshake*. Además, como explicamos a continuación, si el producto no se correspondiera con la descripción proporcionada, el cliente poseería toda la información necesaria para demostrar dicho fraude.

El cliente, hasta que no realiza el pago, no obtiene la clave que le permite descifrar el producto recibido. Conocida esta situación por el vendedor, un vendedor malicioso podría enviar un producto que no fuera el solicitado por el cliente (por ejemplo, otro de menor valor). Sin embargo, si cuando el cliente descifra el producto, éste no coincide con la descripción, el cliente puede presentar al broker la descripción de producto enviada en el mensaje de solicitud de negociación para que compruebe que se corresponde con el resumen digital que el broker recibió en la factura. Junto con la descripción, el cliente también presentará el producto cifrado, para que determine que corresponde con el resumen digital incluido en la factura. En la factura está incluida la clave para poder descifrar el producto. De esta forma, el broker podrá comprobar realmente si el vendedor está cometiendo fraude o no, o si es el cliente el que está presentando falsas acusaciones para desprestigiar la reputación del vendedor.

El cliente también podría intentar adquirir más de un producto con la misma información de pago, lo que se conoce como *doble gasto*. Sin embargo, en el proceso de pago con la tarjeta, el uso de mensajes repetidos se basa en el uso del contador de transacción, que cada vez se incrementa, y en la generación de una clave basada en un reto. En el caso de monedas electrónicas comprobaría el identificador (bien localmente o remotamente, dependiendo del tipo de moneda). En el caso de las tarjetas de crédito/débito este ataque del cliente no tiene sentido porque cada vez le supondría un cargo distinto en su cuenta. Por tanto, este ataque no sería posible.

No repudio

En esta sección analizamos las distintas pruebas de no repudio que obtienen las distintas entidades. Estas pruebas se podrían utilizar en caso de resolución de disputas cuando una parte niegue su implicación en el protocolo o en el envío de cierta información intercambiada.

La primera prueba de no repudio que se genera en el protocolo es la que permite probar que el cliente solicitó un determinado producto y cuál fue el precio acordado por dicho producto. Esta prueba se genera en el mensaje de *Handshake* que el cliente envía al vendedor (junto con el mensaje *NegotiationRequest*).

A continuación, con la factura se puede asociar la descripción del producto solicitado con el producto que envío el vendedor, de forma que, a posteriori, se puede probar que el vendedor envío tal producto asociado a esa factura al cliente. Este mensaje también permite demostrar que el vendedor envió una determinada clave al broker.

Posteriormente, con el mensaje *PaymentOrder*, el broker obtiene una prueba de que el cliente aceptó el pago por un determinado producto de un vendedor concreto.

Finalmente, tanto cliente como vendedor obtienen una prueba de la ejecución exitosa de la transacción por medio del recibo firmado que el broker envía a ambas partes. Este mensaje será utilizado por el cliente para probar, en caso de disputas, la clave recibida.

Intercambio equitativo

El cliente hasta que no realiza el pago no recibe la clave que le permite descifrar el producto. Por otro lado, el cliente cuando realiza el pago se asegura de obtener la clave por la que va a pagar ya que ésta está en posesión del broker. Por tanto, el cliente se asegura de que recibirá la clave para descifrar el producto una vez que haya pagado y el vendedor se asegurar de que el cliente no obtendrá el producto hasta que no haya realizado el pago. Así podemos concluir que el protocolo garantiza esta propiedad.

Atomicidad

En SPEED, antes de que se realice el pago, el cliente tiene el producto cifrado y el broker tiene la clave que permite descifrar dicho producto. De esta forma se evita que el cliente pueda realizar un pago y que el vendedor no le envíe la clave que le permite descifrar el producto. Así, en el protocolo podemos garantizar que no se genera dinero de forma innecesaria y que la transacción ocurrirá completamente (recibiendo cliente y vendedor producto y dinero, respectivamente) o no se produce ningún efecto (el cliente no obtiene el producto y el vendedor no obtiene el dinero). Por tanto podemos afirmar que el protocolo satisface la atomicidad. Como también satisface el intercambio equitativo, podemos garantizar que se cumple la propiedad de buena atomicidad.

Flexibilidad

El protocolo permite realizar el pago con distintos instrumentos de pago, como puede ser un monedero electrónico, una tarjeta de crédito, una cuenta o monedas electrónicas. Además, el protocolo, gracias al posible uso de estos instrumentos, permite pagar importes de diversas cantidades. Por tanto, podemos considerar que el protocolo es flexible y se puede adaptar a distintos escenarios, tal y como veremos en secciones posteriores.

Divisibilidad

SPEED está principalmente pensado para ser utilizado con el monedero electrónico. Sin embargo, como acabamos de mencionar, también soporta otros instrumentos. En los instrumentos descritos (monederos, tarjetas crédito/débito, información bancaria y monedas) se permite ajustar el pago a la cantidad solicitada excepto, quizás, cuando se utilizan monedas electrónicas (dependerá de la propuesta elegida). En general, SPEED ha sido pensado para ser utilizado con monedero. Por tanto, se podría considerar divisible.

Escalabilidad y eficiencia

Tal y como se demostrará en el apartado 3.2.6 el protocolo es eficiente y escalable. Para este propósito, se han realizado diversos tests con distintos números de clientes y distintos tipos de productos.

3.2.5. Implementación

Una vez diseñado SPEED y comprobado que se trata de un protocolo de pagos seguro, decidimos realizar una implementación de éste con el fin de comprobar su viabilidad y su eficiencia. En esta sección comentaremos los detalles de implementación. Posteriormente, en la sección siguiente comentaremos los resultados obtenidos con esta implementación.

La implementación de SPEED se ha realizado de acuerdo a la descripción anteriormente dada. A partir de dicha descripción, se ha realizado una especificación de los distintos elementos del protocolo de acuerdo con la notación ASN.1. Para esta especificación se han utilizado elementos que ya estaban definidos en ASN.1 como los certificados X.509v3 o las distintas estructuras definidas para intercambiar mensajes criptográficos como son PKCS#7/CMS. La especificación del protocolo en ASN.1 se puede encontrar en el Apéndice C.

A partir de esta especificación se realizó un diseño de los distintos componentes del protocolo basado en UML. En el diseño de clases del protocolo se distinguían tres tipos principales de clases: las necesarias para crear los campos de los mensajes, las clases que agrupaban los distintos campos que forman parte de un mensaje y, finalmente, las clases que representaban los mensajes y que realizaban las operaciones criptográficas necesarias para crearlos. En concreto, cada clase de mensajes ofrecía un método denominado *encode* que se encargaba de generar las estructuras ASN.1 a partir de cada uno de los elementos y, a continuación, crear las estructuras PKCS#7/CMS aplicando las operaciones criptográficas necesarias para cada mensaje. Una vez realizadas las operaciones el mensaje ASN.1 se codificaba en DER y así se obtenía una secuencia de bytes que representa al mensaje.

En cuanto a los algoritmos criptográficos utilizados comentar que, para la criptografía asimétrica se utilizaron claves RSA de 1024 bits. Tanto para el cifrado/descifrado simétrico como para la firma basada en HMAC se utilizaron claves IDEA de 128 bits de longitud. Para la función HMAC el algoritmo de resumen digital utilizado fue SHA-1.

Como lenguaje elegido para la implementación de estas clases se utilizó C++ con el compilador *gcc v4.1.2*. Como librería para la creación de los distintos mensajes ASN.1, así como para las distintas funciones criptográficas se ha empleado la librería OpenSSL en su versión 0.9.8g.

3.2.6. Pruebas de rendimiento

Una vez desarrollada la implementación, se llevaron a cabo varios tests para determinar el rendimiento del protocolo. Los distintos experimentos han sido realizados en una red de área local y los equipos utilizados presentan la configuración mostrada en la Tabla 3.2.

Los experimentos se realizaron con distintos tamaños de producto: 32 bytes (lo que podría ser equivalente a una clave de 256 bits), 12 Kb (una página HTML), 281 Kb (equivalente a un documento PDF o una imagen), 3.7 MB (equivalente a una canción un fichero en formato MP3 de corta duración).

Por cada uno de estos tamaños, se usaron simultáneamente hasta cinco ordenadores diferentes actuando como clientes. Por cada tamaño y cada número diferente de clientes se realizaron 100 pruebas.

Características	Valor
Procesador	2 procesadores AMD Opteron(tm) Processor 246
CPU MHz	2000 Mhz
Memoria RAM	2 Gb
Sistema Operativo	Gentoo 4.1.2 p1.1
Kernel	Linux 2.6.24
Compilador	gcc 4.1.2

Tabla 3.2. Configuración de las máquinas para evaluar SPEED.

A partir de estas pruebas se calculó un intervalo de confianza de la mediana al 95% (ver Tabla 3.3). Por ejemplo, el intervalo de confianza al 95% para un cliente que adquiere un producto de 12Kb estaría entre 84,00-4,00 ms y 84,00+4,00 ms) para obtener un valor representativo de cada prueba. De hecho, como se puede comprobar en la Tabla 3.3, en muchos casos el propio valor de la mediana representa el intervalo de confianza. Los clientes realizaban una compra por medio de SPEED en el modo de operación normal y aceptaban el primer precio propuesto por el vendedor. Por tanto, la fase de negociación sólo estaba constituida por tres mensajes. En cuanto a la fase de pago, para las pruebas se supuso que los mensajes del monedero electrónico eran los del monedero WG10, es decir, sería necesario ejecutar al menos una vez el mensaje *SAMmessage* y el mensaje *PurseMessage*.

	32 bytes	IC 95%	12 Kb	IC 95%	281 Kb	IC 95%	3.7 Mb	IC 95%
1 cliente	80,00	±0,00	84,00	±4,00	124,00	±0,00	591,99	±0,00
2 clientes	88,00	±0,00	88,00	±0,00	136,00	±0,00	768,00	±22,19
3 clientes	88,00	±0,00	88,00	±0,00	132,00	±8,00	907,99	±3,97
4 clientes	86,00	±2,00	88,00	±0,00	136,00	±0,00	1236,00	±0,00
5 clientes	88,00	±0,00	92,00	±4,00	144,00	±4,00	1539,96	±8,02

Tabla 3.3. Tiempos en milisegundos e intervalos de confianza para SPEED.

Como muestra la Figura 3.3 (creada a partir de la Tabla 3.3), el protocolo se comportaba de manera adecuada con varios clientes realizando compras a la misma vez. Es importante mencionar que por simplificar la figura, las distintas categorías de tiempos no están a escala con respecto al tamaño del producto. Sin embargo, podemos afirmar que el crecimiento de los distintos tiempos con respecto a este tamaño es lineal. En el tiempo de la compra inciden dos elementos: por un lado el tiempo relacionado con la entrega del producto y, por otro lado, el tiempo relacionado con el resto de los mensajes. El mensaje

de entrega del producto es el que origina las diferencias entre aquellas compras del mismo tamaño de producto; sin embargo, el tiempo para intercambiar el resto de los mensajes permanece casi independiente del número de clientes concurrentes, tal y como se muestra en los resultados obtenidos en la compra del producto cuyo tamaño es 32 bytes.

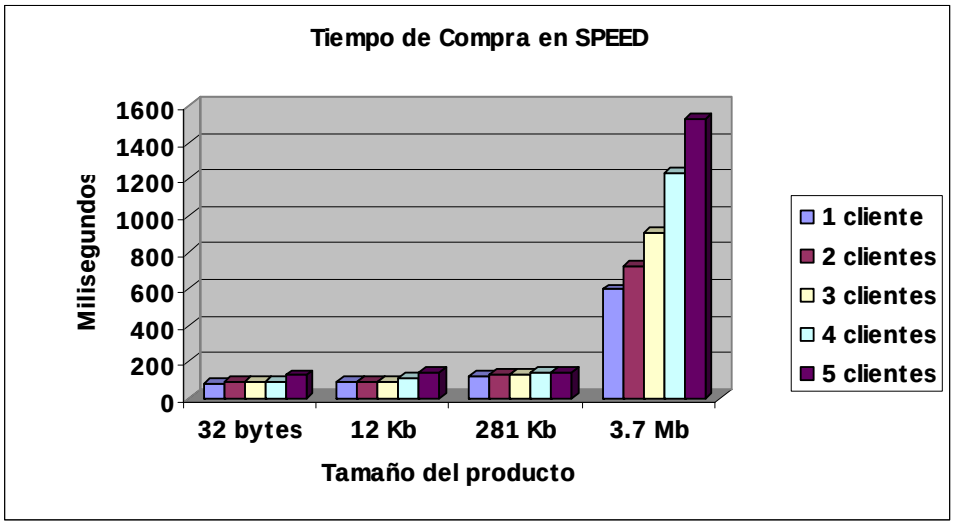


Figura 3.3. Tiempo de Compra en SPEED.

En las siguientes secciones describiremos tres escenarios donde este protocolo se ha utilizado para realizar pagos. En cada uno de estos escenarios o entornos de aplicación explicaremos cómo se ha aplicado el uso de SPEED, para qué productos y si ha sido necesario el desarrollo de nuevos componentes.

3.2.7. Escenarios de validación de SPEED

En esta sección presentamos tres escenarios que han sido utilizados para validar el protocolo de pagos que acabamos de introducir. De cada uno de estos escenarios comentaremos las características más destacadas así como indicaremos bajo qué circunstancias se ha utilizado SPEED.

El proyecto PISCIS

El *Proyecto Piloto de definición de una Infraestructura de Seguridad para el Comercio Inteligente de Servicios* (PISCIS) [37, 113, 114] surgió de la colaboración entre los grupos de investigación de dos universidades, la Universidad Politécnica de Madrid y la Universidad de Murcia, con la empresa proveedora de servicios de redes de cable ONO bajo el marco de los proyectos FEDER. El objetivo de este proyecto era el desarrollo de un piloto que permitiera a los usuarios apreciar las ventajas del comercio electrónico. Para este fin, se diseño e implantó una infraestructura de seguridad sobre la que crear un

sistema de comercio electrónico seguro basado en los últimos avances en seguridad disponibles en ese momento.

En el proyecto, los principales componentes que se desarrollaron fueron: una infraestructura de certificación avanzada, una serie de componentes para garantizar la seguridad de las transacciones Web por medio de tarjetas inteligentes y, por último, el protocolo de pagos SPEED para su uso en la Web.

Todos estos componentes fueron utilizados en un entorno de pruebas real que incluía la conexión entre la Universidad de Murcia y la empresa de servicios a través de cable ONO, un portal de comercio electrónico y un demostrador conocido como *Gramola Virtual*.

En este escenario, uno de los principales componentes era el protocolo de pagos SPEED que permitía la compra de los productos por medio del monedero de la tarjeta inteligente que, en este caso, era un monedero WG10. El pago se realizaba con una serie de módulos SAM distribuidos conectados al broker [115].

Finalmente, la denominada *Gramola Virtual* consistía en un dispositivo cerrado que estaba pensado para ser implantado en lugares de ocio, como bares, restaurantes, pubs, etc., en los que los usuarios podían adquirir las canciones que se ofrecían en el portal de comercio a través de la red de ONO. Básicamente, la Gramola Virtual ejecutaba el navegador Web en modo quiosco y permitía al usuario interactuar única y exclusivamente con el portal de comercio. Este portal permitía la compra de música en formato MP3 y para el proceso de pago se basaba en el protocolo SPEED. Para la parte cliente, para la integración de SPEED en el navegador se desarrollaron una serie de ActiveXs que utilizaban la implementación presentada anteriormente para realizar las pruebas de rendimiento. Para la parte del servidor se desarrolló una nueva implementación de SPEED basada en Java. Este escenario permitió mostrar el uso del protocolo así como la interoperabilidad de dos implementaciones distintas a partir del uso de ASN.1.

Fidelización electrónica

En los entornos de B2C una de las principales preocupaciones de las empresas es mantener una relación activa y duradera con sus clientes. Una forma de retener e incrementar las compras de los clientes es por medio de los programas de fidelización. Mediante este tipo de programas tanto los usuarios como los comerciantes obtienen beneficios. Por un lado, los clientes obtienen mejores precios, ofertas especiales, promociones, facilidades de pago, etc. Por otro lado, los comerciantes consolidan sus clientes, mantienen un alto grado de fidelidad, consiguen un mayor número de transacciones, mayores beneficios, menores costes a la hora de conseguir nuevos clientes, etc. [235].

De entre los distintos mecanismos de fidelización, uno de los más habituales es la pertenencia a un determinado grupo beneficiario. Esta pertenencia se consigue por medio de una cuota que el usuario tiene que satisfacer. En general, dentro del mismo servicio, suelen existir distintos grupos beneficiarios, cada uno de los cuáles suele tener beneficios distintos, siendo mejores a medida que la cantidad a satisfacer como cuota es mayor.

Con el objetivo de soportar este escenario de fidelización, se desarrolló un sistema de suscripción electrónica basado en dos componentes fundamentales. Por un lado, el protocolo de pagos SPEED ya presentado en este capítulo para el pago de la pertenencia a una suscripción como para el pago basado en suscripción. Por otro lado, la especificación SPKI/SDSI para especificar las distintas relaciones de pertenencia de los usuarios a los grupos suscritos, así como para representar las ventajas asociadas a dicha pertenencia. Asociado a la especificación SPKI/SDSI utilizaremos el sistema para la gestión distribuida de certificados SPKI, denominado DCMS (Distributed Credential Management System) [36].

La especificación SPKI/SDSI [87] define distintos tipos de certificados para la asignación de identidades y para autorizaciones. El DCMS define cómo deben expresarse las solicitudes de certificación para los distintos tipos de certificados, cómo se satisfacen las políticas de seguridad, cuáles son las identidades participantes en un proceso de certificación y, finalmente, cómo estas entidades intercambian información relacionada con el proceso de autorización. Más detalles acerca de los distintos mecanismos se pueden encontrar en [36].

El sistema de suscripción electrónica desarrollado, en concreto, está aplicado al campo de la venta de billetes de avión. El modelo genérico en el que está basado esta propuesta se encuentra descrito en más detalle en [38]. Los distintos componentes aparecen reflejados en la Figura 3.4 y serán descritos a continuación.

En el entorno propuesto, las agencias de viajes ofrecen a sus clientes la suscripción a distintos grupos. En este caso los grupos son: oro, plata y joven. La pertenencia al grupo oro es la que supone un importe mayor, sin embargo, dará mejores privilegios que los otros grupos. En este escenario, los privilegios consisten en el descuento de un determinado porcentaje previamente establecido sobre el precio de los billetes de vuelo de una determinada compañía aérea.

Cada compañía aérea, a la hora de realizar la venta de sus billetes, permite la presentación de credenciales de pertenencia a grupos con el fin de que estos clientes que pertenecen a un determinado grupo obtengan un descuento en el precio de billete.

Con el fin de facilitar a los usuarios la gestión y almacenamiento de tanto las credenciales de grupo como los billetes electrónicos adquiridos, se incorporó al sistema el uso de tarjetas inteligentes Java Card. Gracias al uso de estas tarjetas para almacenar credenciales y billetes se facilita que el usuario pueda utilizarla en distintos terminales y, por tanto, facilitando su movilidad.

En este escenario hay dos procesos fundamentales que se describirán a continuación. Por un lado, el proceso de suscripción a una agencia de viajes. Por otro lado, el proceso de compra de billetes. Ambos procesos se realizan vía Web.

En el proceso de suscripción (marcado como 1 en la Figura 3.4), un usuario se conecta al portal Web de un gestor de subscripciones en el que puede consultar las distintas agencias que ofrecen pertenencia a grupos, así como los privilegios de cada grupo. Estos privilegios están expresados como certificados de atributos SPKI/SDSI.

Si el usuario decide subscribirse al grupo, se procede al inicio del protocolo SPEED para realizar el pago y así obtener un certificado de pertenencia al grupo elegido. Este certificado será almacenado en la tarjeta Java Card.

En este caso, el gestor de suscripciones actúa, por un lado, como vendedor, ya que recibe el pago por un producto que es un certificado y, por otro lado, actúa como autoridad de nombramiento al emitir un certificado que liga la clave pública del usuario con un determinado grupo de usuarios.

En el proceso de compra de billetes (marcado como 2 en la Figura 3.4) el usuario se conecta al portal Web de una organización que realiza la venta de billetes de distintas compañías.

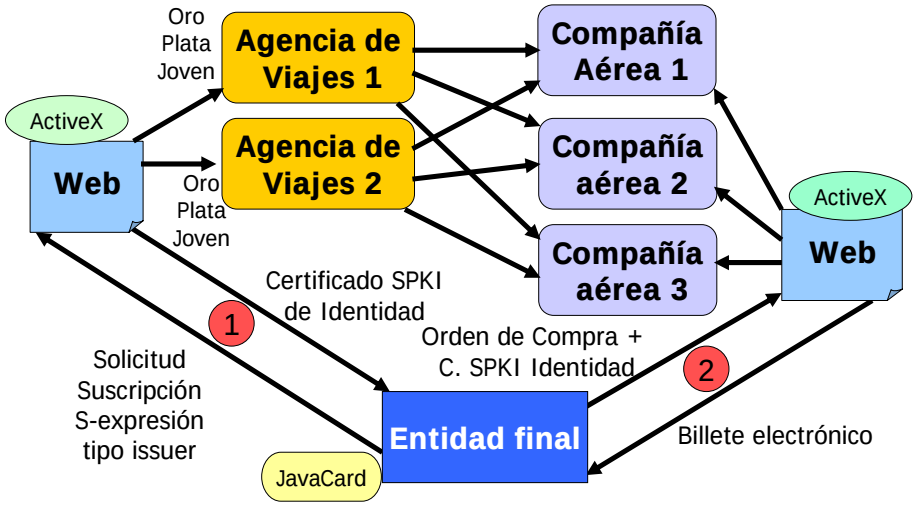


Figura 3.4. Escenario de suscripción electrónica.

Una vez que el usuario elige los detalles del vuelo concreto (fecha y hora), se inicia la fase de pago que, en primer lugar, supone el acceso a la tarjeta Java Card para la obtención de los certificados de identidad (que liga al usuario a un determinado grupo) y los certificados de atributos (que indican los distintos descuentos a aplicar al grupo). A continuación se elige, en función de las características del vuelo, el certificado de identidad que proporcionará un mayor descuento en la compra del billete. Una vez realizada la elección, se inicia el protocolo de pago SPEED.

El servidor de venta de billetes cuando recibe el mensaje inicial de SPEED obtiene las credenciales proporcionadas y realiza las verificaciones relacionadas con el billete solicitado así como ejecuta el algoritmo de cálculo de autorizaciones para determinar el descuento a aplicar (la descripción del algoritmo utilizado viene descrito en más detalle en [86, 87, 111]). Si el usuario está de acuerdo con el precio final, entonces continúa la ejecución del protocolo SPEED para obtener el billete solicitado que estará firmado por la compañía emisora.

Finalmente, el billete electrónico obtenido se almacena en la tarjeta Java Card. De esta forma, el usuario posteriormente podrá presentar su billete como ticket de embarque en el vuelo deseado.

Así, en este escenario hemos mostrado como el protocolo de pagos SPEED por un lado se utiliza para el pago de la suscripción a un grupo y, por otro lado, se ha explicado cómo el protocolo se utiliza para la compra de billetes electrónicos a la misma vez que permite el envío de las credenciales obtenidas por el usuario con el fin de obtener ventajas adicionales, como es el descuento en el precio del billete electrónico solicitado.

Gestión de Derechos de Autor

Actualmente, uno de los mayores problemas a los que se enfrentan autores, editores y distribuidores de contenido digital es la facilidad con la que este contenido puede ser copiado sin realizar el pago correspondiente a los derechos de autor asociados al uso y disfrute de la obra.

La gestión de derechos digitales (DRM – Digital Rights Management) se propone con el objetivo de negociar y controlar los derechos de propiedad intelectual (IPR – Intellectual Property Rights) asociados a material con copyright. Relacionado con esta gestión de los derechos digitales se propuso SECURingIPR (SECure infrastructure for negotiation and purchase of Intellectual Property Rights) [250] como un sistema DRM genérico donde el modelo de negocio estaba basado en el desarrollo de una arquitectura para negociar, alcanzar acuerdos y comprar, de una forma segura, contenido sujeto a derechos de autor.

Este modelo garantiza que el flujo de información y derechos desde los creadores hasta los usuarios finales se realiza de una forma segura, utilizando para el intercambio de la información de derechos de autor un lenguaje de derechos de autor (DRL – Digital Rights Language) como Open Digital Rights Language (ODRL) [314]. El modelo genérico se muestra en la Figura 3.5. A continuación pasamos a describir este modelo de forma breve. Una descripción más detallada se encuentra [249].

En SECURingIPR, una obra que llega al usuario final procedente de un creador sigue el flujo que comentamos a continuación. El ciclo de vida se inicia con la firma de un contrato entre el creador (u otra entidad que lo represente, llamada gestor de los derechos) y el editor. Este contrato recibe el nombre de *contrato de autor*. Con este contrato, el creador negocia las condiciones y los derechos bajo los cuales la obra será editada y distribuida así como los beneficios que obtendrá de los derechos asociados a la obra. A la misma vez, el editor obtiene los derechos necesarios para iniciar la producción de la obra. Posteriormente, el editor llegará a un acuerdo con uno o más distribuidores para que ésta llegue a los usuarios finales y puedan adquirirla. Este acuerdo de distribución se refleja en un contrato llamado *contrato de distribución*. En este contrato, los derechos de la obra se distribuyen y se define el beneficio que obtendrá de tales derechos cada parte.

Este escenario está basado en el uso de dos protocolos. Por un lado, el protocolo SPEED permite negociar el precio del producto, realizar el pago y la entrega del producto de una forma segura. Por otro lado, el protocolo de firma de contratos SURENESS [248]

se utiliza para todos los procesos relacionados con la negociación y firma de contratos entre las distintas partes.

En la ejecución de este modelo, tanto para SURENESS como para SPEED, los productos y derechos asociados son expresados por medio de expresiones ODRL. Este lenguaje contiene además expresiones para llevar a cabo la negociación. De esta forma, tanto en SURENESS como en SPEED las partes pueden realizar ofertas y contra-ofertas hasta alcanzar un acuerdo.

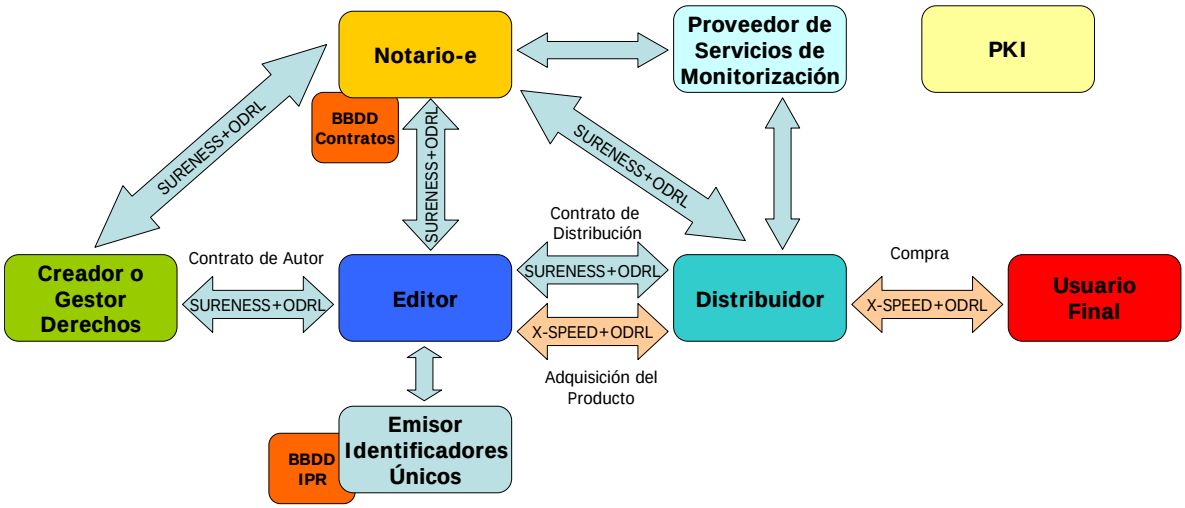


Figura 3.5. Implementación de SECURingIPR.

Para el escenario descrito se desarrollaron tanto los protocolos SPEED y SURENESS diseñando e implementando todos los mensajes en XML de acuerdo a sus definiciones. Los formatos de cifrado y de firma escogidos fueron XML Signature y XML Encryption, respectivamente. El motivo del desarrollo de esta nueva implementación de SPEED obedece a dos causas. En primer lugar, probar los estándares XML Signature [312] y XML Encryption [313] y comprobar que SPEED puede ser implementado con tales estándares. Segundo, puesto que cada vez más la información relacionada con el comercio electrónico está expresada en XML, parece natural que los protocolos que intercambian esta información también estén expresados en XML. A esta nueva implementación de SPEED se la ha denominado *x-SPEED* para reflejar el hecho de que los distintos mensajes han sido codificados en XML. Para esta implementación, se ha utilizado Java como lenguaje de programación. En cuanto a las librerías utilizadas podemos mencionar: Xerces, para el procesamiento de los datos en XML, XML Security Suite de IBM, para los formatos de firma y cifrado en XML y, finalmente, IAIK como proveedor criptográfico.

Basados en el escenario descrito y en los componentes mencionados, se ha desplegado un sistema de venta de vídeos musicales para el que se ha desarrollado una aplicación Web basada agentes inteligentes. Una vez que el cliente ha elegido un vídeo, a partir de las preferencias del usuario (precio mínimo, precio máximo, número de reproducciones, etc.), será un agente el que negocie y compre los derechos para reproducir tal canción. En

este escenario una vez que el pago se ha realizado, el usuario puede reproducir el contenido.

En este modelo, para desalentar la distribución de contenido ilegal, el proveedor de contenido podría incrustar una marca de agua en cada contenido distribuido. Para proteger los intereses tanto de compradores como de vendedores se han propuesto dos tipos de esquemas de protección de la distribución del contenido. Estos esquemas son: Buyer-Seller Watermarking (BSW) [189], un esquema ligero de BSW [315] y Asymmetric Fingerprinting (AF) [222]. Incluso se han propuesto las versiones anónimas, para proteger la identidad de cliente, de ambos esquemas: [34, 55, 167, 281]. En los esquemas basados en BSW, una TTP genera las marcas de agua para el comprador, mientras que en los esquemas AF el comprador genera su propia marca de agua. En ambos esquemas se evita que un comprador honesto sea incriminado injustamente, así como que un cliente deshonesto pueda negar que haya distribuido ilegalmente tal contenido.

En este escenario hemos podido comprobar el uso de una nueva implementación de SPEED con estándares XML. Esta implementación nos ha permitido realizar la negociación y compra de contenidos electrónicos y sus derechos asociados.

3.2.8. Conclusiones

A lo largo de esta Sección 3.2 hemos presentado SPEED, un protocolo de pagos que garantiza la seguridad en todas las fases que pueden tener lugar en la compra de un producto electrónico: negociación, pago y entrega. De las propiedades de seguridad hemos realizado un análisis pormenorizado. Además, la validación de estas propiedades de seguridad ha sido realizada por medio de una herramienta de validación automática como AVISPA [9, 11, 303].

La solución presentada incorpora como un elemento fundamental el uso de la tarjeta y su monedero inteligente, dotando así al sistema de flexibilidad a la hora de realizar el pago, tanto en escenarios on-line como en escenarios locales donde existe una máquina de vending. La flexibilidad también se manifiesta en la posibilidad de que los monederos pueden realizar el pago de distintos tipos de importes. Además, facilita a los vendedores la aceptación de pagos basados en este tipo de dispositivos ya que el soporte de los distintos tipos de módulos SAM lo proporciona un tercero de confianza.

Aunque el protocolo se ha diseñado para realizar el pago principalmente con tarjeta inteligente, también sería posible utilizarlo para el pago de monedas electrónicas o por medio de cuenta o tarjeta de crédito/débito, de forma que se ofrece un protocolo útil para distintos tipos de escenarios.

Esta propuesta, lejos de quedarse sólo en su diseño ha sido implementada, testeada y evaluada. Los resultados muestran que el sistema es implementable, que se puede aplicar a distintos escenarios y que es escalable para la adquisición de productos electrónicos. Por tanto, con SPEED ofrecemos una solución que es segura, eficiente, flexible y cumple los requisitos exigidos a los protocolos de pago definidos en el Capítulo 2.

Además, esta solución, combinada con las técnicas de gestión de derechos de autor descritas en la anterior sección, permitiría que las entidades que participan en el ciclo de

vida de los contenidos digitales puedan recibir las cantidades correspondientes a los derechos de autor que poseen. Así, esta solución, desde el punto de vista de la seguridad, puede ayudar a diseñar y crear infraestructuras seguras que se enfrenten a los problemas de distribución ilegal de contenidos que actualmente está siendo ampliamente estudiado en la comunidad científica y en la industria musical y cinematográfica.

Finalmente, esta propuesta combinada con un framework de pagos puede ofrecer a los usuarios un mayor nivel de confianza a la hora de realizar transacciones de compra en la Web.

3.3. Monedero electrónico basado en la generación de monedas (PURSE-COIN)

Las tarjetas inteligentes que actúan como monedero electrónico son más seguras que las tarjetas de crédito y se utilizan habitualmente para realizar pagos en máquinas de vending o en puntos de venta, tal y como mencionábamos en la Sección 2.2.3. Sin embargo, a pesar de este hecho, su uso no está extendido para el pago de productos por Internet.

A día de hoy, si un vendedor quiere vender sus productos ofreciendo como método de pago el monedero electrónico tiene que integrar en su aplicación de comercio electrónico dispositivos o terminales punto de venta que incluya un módulo SAM para llevar a cabo la autenticación entre el monedero y éste dispositivo (ver sección 2.2.4), tal y como se muestra en la Figura 3.6.

Esta integración no es inmediata y las transacciones son lentas debido al intercambio de mensajes que es necesario entre monedero y el módulo SAM a través de una conexión HTTP/TCP. El problema de la integración para el vendedor se puede solucionar con SPEED. Aún así, SPEED no evita la comunicación necesaria entre el monedero y el módulo SAM.

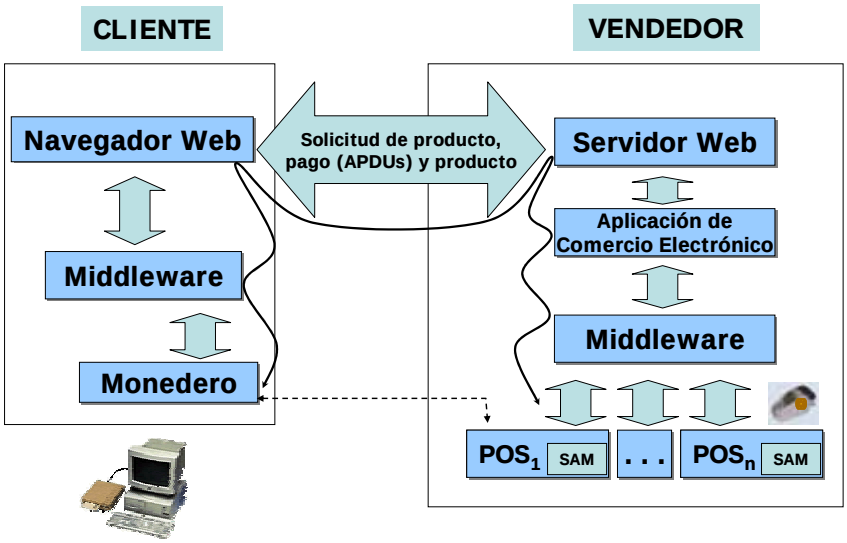


Figura 3.6. Pago en Internet con monedero electrónico.

Por otro lado, las monedas electrónicas (ver Sección 2.2.2) son más adecuadas para llevar a cabo pagos en la red porque están representadas como una secuencias de bytes que puede ser fácilmente transportadas como parte de la información de compra. Además, éstas pueden ser verificadas vía software y/o mediante el establecimiento de una conexión en línea con el emisor y, por tanto, no es necesario el uso de ningún hardware específico. En general, como mencionábamos en la Sección 2.3.1, los esquemas de moneda electrónica se puede clasificar en dos grandes grupos dependiendo de si éstas pueden ser utilizadas con más de un vendedor: los de moneda genérica, que pueden ser utilizados con cualquier vendedor (como por ejemplo eCash, Paycash, etc.), y los de moneda específica, donde las monedas sólo son válidas para un determinado vendedor (como por ejemplo Millicent, Payword, etc.).

La principal ventaja de las monedas genéricas es que éstas pueden ser utilizadas con cualquier vendedor. La principal desventaja es que, en general, se requiere que el vendedor verifique en línea las monedas con el emisor para estar seguro de que la moneda no fue previamente utilizada (tal y como se muestra en la Figura 3.7) y así evitar problemas de doble gasto. Aunque, como se mencionó en la Sección 2.2.2, existe una propuesta de moneda genérica que no necesita verificación en línea a la hora de realizar el pago, si bien ésta presenta el problema de la divisibilidad, ya que las monedas son de un determinado importe específico. Así, en el caso de que el cliente no tenga una moneda del importe por el que desea realizar el pago, tendrá que conectarse al banco para obtenerla.

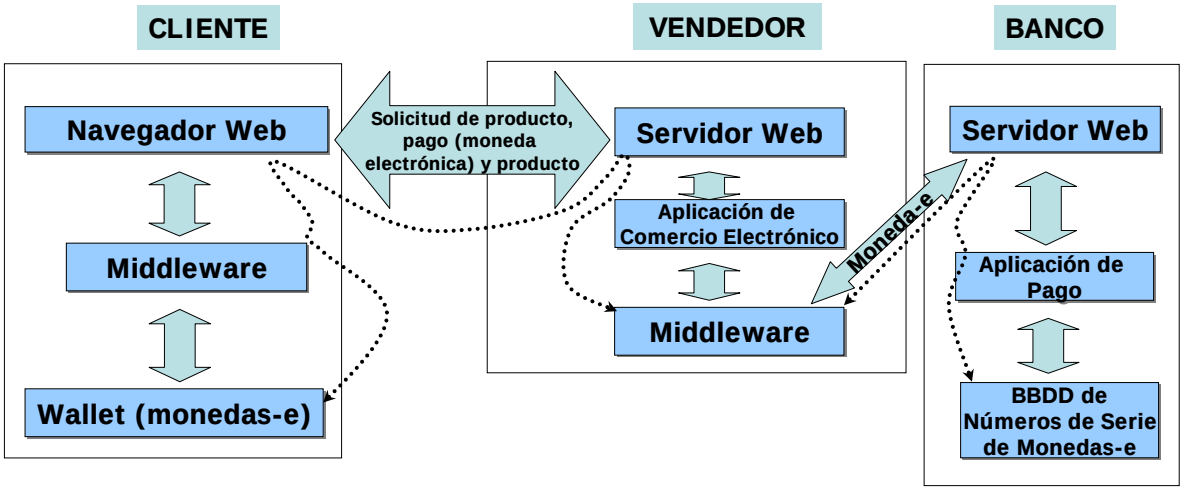


Figura 3.7. Pago en Internet con moneda electrónica.

Por otro lado, las monedas que son específicas para un determinado vendedor permiten un mejor control del doble gasto ya que pueden ser controladas por el vendedor y, por tanto, no es necesario establecer ninguna conexión con una tercera parte. El principal problema se presenta para el usuario que tiene que manejar dinero electrónico de cada uno de los distintos vendedores con los que desee efectuar pagos. Así, el usuario podría tener importantes cantidades de dinero electrónico con distintos vendedores que no podría utilizar con otros o que no podría ser dividido en cantidades inferiores.

Como respuesta a esta problemática, en esta sección presentamos un nuevo método de pago denominado PURSE-COIN [251, 252]. Su nombre se debe a que combina las ventajas de los monederos electrónicos y de las monedas electrónicas. Es decir, por un lado es seguro y portable como un monedero, ya que se trata de una aplicación de pago almacenada en una tarjeta Java Card (ver Sección 2.2.3). Así, como cualquier otro sistema de monedero electrónico, el dinero se puede gastar con cualquier vendedor y en la cantidad deseada (siempre que el saldo sea suficiente). Por otro lado, durante el proceso de pago, el monedero no intercambia mensajes con un módulo SAM sino que genera una moneda electrónica que es específica para el vendedor al que realiza el pago. De esta manera, la moneda electrónica puede ser verificada sin necesidad de establecer una conexión en línea con el banco. Además, el vendedor no necesita instalar ningún hardware específico para recibir las monedas del monedero.

Debido al hecho de que esta propuesta está basada en un applet Java Card, ésta puede ser desarrollada tanto en tarjetas inteligentes tradicionales (de acuerdo con el tamaño especificado en el estándar ISO 7816 [142]) como en tarjetas SIM. Por tanto, incluso esta propuesta se podría considerar como la base de un sistema de pago móvil cuyas monedas podrían ser transportadas con cualquiera de los mecanismos de transporte soportados por los dispositivos móviles.

3.3.1. Diseño del sistema

El modelo de negocio de PURSE-COIN está basado en prepago y las entidades participantes son: el cliente, el vendedor y el emisor de monederos electrónicos. Éste último normalmente se trata de una entidad financiera pero también podría ser un proveedor de servicios de pago (PSP) o incluso un operador de telefonía.

En este modelo, un requisito esencial es que los clientes posean una tarjeta inteligente Java Card (ver Sección 2.2.3) que facilite la incorporación de aplicaciones después de su emisión, con un monedero electrónico desarrollado como un applet (que puede estar preinstalado o podría ser incorporado después de la emisión de la tarjeta), que contiene una clave privada y un certificado de monedero electrónico como la base para llevar a cabo pagos. Cabe destacar que esta clave nunca saldrá de la tarjeta inteligente y que cualquier operación requiere que el usuario se autentique a la tarjeta por medio de la introducción de un número de identificación personal o número PIN (Personal Identification Number) que sólo éste debe conocer.

La idea de la que parte nuestra propuesta es que nuestro monedero, en lugar de intercambiar APDUs con un módulo SAM, generará monedas electrónicas específicas de un vendedor particular. Las monedas son generadas por el cliente enviando algunos APDUs al monedero, bien por medio de un lector de tarjetas inteligentes o bien por medio de un dispositivo móvil (en el caso de un teléfono móvil con una tarjeta SIM). En estos casos, la seguridad del canal de comunicaciones utilizado para enviar estas APDUs está gestionada completamente por el dispositivo que lo controla y, como se menciona en [187, 234], podría ser considerado seguro. De esta manera, con las monedas electrónicas, los vendedores sólo necesitan verificar firmas digitales y los correspondientes certificados para

aceptar pagos. El certificado de monedero se verifica contra un conjunto de certificados raíces que son considerados emisores de monederos de confianza y que el vendedor tiene instalados. Por tanto, los vendedores no necesitan ni un dispositivo SAM ni una conexión on-line con el emisor de tarjetas para cada pago.

En este modelo, hay dos entidades que pueden generar monedas electrónicas: el emisor de tarjetas, para incrementar el saldo de la tarjeta, y el monedero electrónico para realizar pagos a los vendedores.

Fase de operación

En esta sección explicamos las distintas fases que tienen lugar en el ciclo de vida del monedero PURSE-COIN. Estas fases y sus mensajes se explicarán en más detalle en las siguientes secciones.

Cuando el usuario obtiene el monedero, si la clave privada y el certificado no han sido previamente instalados, éste tiene que llevar a cabo un proceso de certificación como aparece en la fase I de la Figura 3.8 (certificación de monedero). Este proceso es similar a solicitar un certificado a una CA.

Sin embargo, antes de poder utilizar el monedero, es preciso cargarlo con dinero electrónico. A este proceso se le denomina carga del monedero electrónico (en la Figura 3.8 se corresponde con la fase II).

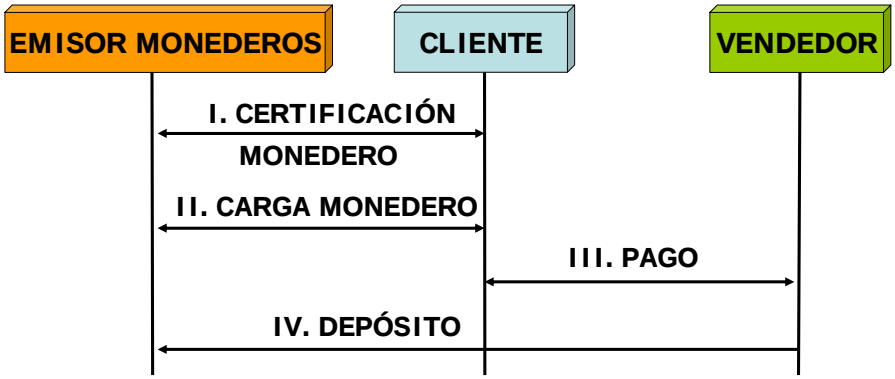


Figura 3.8. Modelo de Negocio.

Cuando un cliente quiere realizar un pago, el monedero tiene que generar una moneda específica de vendedor de acuerdo con la cantidad deseada y siempre que ésta no supere el saldo disponible en la tarjeta. Una vez generada la moneda, el cliente la envía al vendedor por medio de un protocolo acordado entre ambos. A continuación, una vez que el vendedor verifica el pago, éste procede a enviar o proveer el producto o servicio solicitado. Estos pasos se corresponde con la fase III de la Figura 3.8 (pago).

Posteriormente, cuando el vendedor estime oportuno o cuando haya acordado con el emisor de tarjetas (podría ser al final del día, semanalmente, etc.), envía las monedas recibidas al emisor de tarjetas (de aquí en adelante, nos referiremos a esta entidad simplemente como emisor) para recibir el correspondiente pago al valor de dichas monedas

(fase IV, depósito). Este emisor lleva a cabo varias tareas: recepción de las monedas de los vendedores, reembolso de las monedas recibidas por los vendedores a sus respectivas cuentas, carga de monederos, operaciones relacionadas con los certificados (emisión, revocación, etc.) y detección de posibles fraudes.

3.3.2. Características del monedero

El monedero almacena la siguiente información:

- **Número de Identificación Personal** (conocido como número PIN). Este número representa un código personal que se requiere para iniciar cualquier operación de pago. Este valor sólo puede ser comprobado y su valor nunca saldrá de la tarjeta inteligente. Como medida de seguridad, si un cliente falla tres veces consecutivas en el proceso de autenticación, el monedero se bloqueará.
- **Número de Serie** (SN_{EP}). Representa una secuencia de bytes identificando de manera única al monedero.
- **Clave simétrica diversificada** (DSK – Derived Symmetric Key). Esta clave se obtiene a partir de una clave maestra y el identificador del monedero. Este proceso aparece descrito en [44]. La clave maestra sólo es conocida por el emisor.
- **Saldo**. Es un contador que almacena la cantidad actual disponible para realizar pagos con el monedero.
- **Contador de transacciones**. Este contador almacena el número de operaciones realizadas por el monedero, es decir, el número de transacción (TN_{EP}). Este contador se incrementa cada vez que el monedero efectúa una operación.
- **Clave privada**. Es una clave asimétrica que soporta operaciones de firma digital, como podría ser RSA, DSA, ECDSA, etc.
- **Certificado de monedero**. Este certificado lo genera el emisor y es conforme al formato X.509v3.
- **Certificado del emisor**. Este certificado, también X.509v3, permite comprobar la información recibida del emisor.

El monedero, aparte de almacenar la información que acabamos de describir, también proporciona las operaciones necesarias para la gestión de las claves privadas, certificados, así como todas las operaciones (consulta, actualización, etc.) relacionadas con los campos que acabamos de mencionar.

3.3.3. Certificación del monedero

Cuando un cliente obtiene un monedero, éste podría tener (o no) pre-instalada una clave privada y un certificado de monedero. Si tiene una clave y un certificado, el monedero está preparado para obtener dinero, es decir, para realizar una carga y para

realizar pagos. En caso contrario, es necesario que genere una clave privada y después obtenga un certificado de monedero.

La decisión de incluir o no la clave pre-instalada dependerá del emisor y de su proceso de generación y distribución de monederos. El realizar la generación la clave privada y la emisión del certificado asociado durante la personalización del monedero conlleva que este proceso sea más complejo y lento. Por otro lado, evita que se tenga que llevar a cabo un proceso de certificación inicial.

La principal razón por la que decidimos diseñar este proceso de certificación se debe a que queremos estar seguros de que el emisor está realmente certificando a un monedero electrónico y no a cualquier otra clave privada que esté fuera de éste. Si el emisor emitiera un certificado a cualquier otra clave privada, el usuario podría generar tantas monedas como quisiera sin tener que pagar por ellas. De esta manera, se permite que la generación de la clave privada se lleve a cabo de forma posterior a la distribución del monedero.

El proceso de obtención de un certificado es similar al seguido por un usuario para obtener un certificado de identidad. La principal diferencia entre obtener un certificado de identidad y un certificado de monedero radica en que, para realmente certificar la clave generada en un monedero (y no cualquier otra clave), el proceso está basado en el uso de una clave diversificada (DSK) que fue establecida en el monedero cuando el applet de monedero fue instalado en la tarjeta inteligente.

En este proceso, es importante destacar que el certificado emitido no está asociado a ninguna entidad, sólo al identificador del monedero. Así, inicialmente, los pagos son anónimos como un monedero electrónico tradicional (ver Sección 2.2.4). En este proceso de certificación los pasos a seguir son:

1. El emisor de monederos envía un reto a la tarjeta.

```
1.1 EPI → C: Challenge
```

donde:

- *EPI (Emisor de Monederos)*. Representa al emisor del monedero electrónico.
- *C* (*Cliente*). Representa al cliente.
- *Challenge (Reto)* es un array de bytes generado aleatoriamente.

2. El monedero genera una clave privada basada en criptografía asimétrica. Para este propósito, el monedero recibe una APDU de generación de clave privada.

```
2.1. C → EP: AP(generate asymmetric key:rsa,2048).
2.2. EP → C: APD(R_CODE,[EPPKey]).
```

donde:

- *EP* (*Monedero*). Representa al monedero.

- *AP(X:Y)*. Representa a una APDU que indica que se quiere ejecutar el comando correspondiente a la operación *X* con los parámetros *Y*. Los distintos parámetros aparecerán separados por comas. Por tanto, *AP(generate asymmetric key:rsa,2048)* representa la APDU que se envía al monedero pera generar una clave asimétrica de tipo RSA cuya longitud es 2048 bits.
- *APD(D)*. Representa una APDU que contiene los datos especificados por *D*.
- *R_CODE*. Indica el resultado de la ejecución de una determinada operación.
- *EPPKey (Clave pública del monedero)*. Corresponde con la clave pública asociada a la clave privada generada en el caso de que la operación se haya realizado correctamente.

3. El monedero utiliza la clave simétrica diversificada (DSK) para firmar la solicitud de certificación que será enviada al emisor. La información "firmada" junto con un reto permite al emisor autenticar al monedero. A diferencia de una PKI (Public Key Infrastructure – Infraestructura de Clave Pública) donde la solicitud está en formato PKCS#10 [246] y está firmada por la clave privada que se acaba de generar, en nuestra propuesta, la solicitud se firma con la clave simétrica. A continuación, detallamos este proceso:
   a. El emisor envía un reto (*Challenge*) al monedero. Este reto es una serie de bytes que han sido generados aleatoriamente.

      3.1. EPI → C: Challenge

   b. El monedero envía al módulo del emisor de tarjetas inteligentes o EM su clave privada firmada por medio de una función HMAC con la clave DSK que está almacenada en la tarjeta.

      3.2. C → EP: APDU(sign e-purse key:Challenge)
      3.3. EP → C: APD(SN_{EP}, SR, EPPKey)
      3.4. C → EPI: SN_{EP}, SR, EPPKey

   donde:
   - La función *sign e-purse key* (*firma clave monedero)* se encarga de firmar con la clave diversificada existente en el monedero (DSK), el reto recibido como parámetro junto con la clave pública asociada a la clave privada existente en el monedero.
   - $SR = H_{DSK}(Reto, EPPKey)$

4. El EPI envía la información recibida al módulo SAM_{EPI} para que calcule la clave DSK y comprueba si la clave privada fue firmada correctamente y, por tanto, la

solicitud realmente proviene de un monedero. En este caso, el módulo SAM_{EPI} emite el certificado de monedero.

4.1. EM → SAM_{EPI}: APDU(verify e-purse key: SN_E,SR,EPPKey,Reto)
4.2. SAM_{EPI} → EM: APD($Cert_{EP}$)

5. El certificado emitido (*$Cert_{EP}$*) se envía al monedero. Una vez recibido, el monedero verifica el certificado por medio de la clave pública del emisor. Si la verificación es correcta, el monedero almacena el certificado.

5.1. EPI → C: $Cert_{MP}$
5.2. C → EP: APDU(Install certificate:$Cert_{EP}$)
5.3. EP → C: APD(R_CODE)

donde:

- *Install Certificate (Instalar certificado)* es una función que permite instalar el certificado recibido en el monedero.

3.3.4. Carga del monedero

El emisor puede ofrecer distintos métodos de pago para realizar el cargo o pago que permitirá incrementar el saldo del monedero como, por ejemplo, tarjetas de crédito, transferencias bancarias, etc.

En este proceso de pago, el usuario, además de la información de pago, envía determinada información que permite evitar ataques de reenvío y que podría permitir incrementar varias veces el saldo del monedero de forma no permitida.

Una vez que el pago se ha efectuado, el emisor acuña una moneda con la información recibida. Esta moneda, que se muestra en el paso 4 del intercambio de mensajes que figura más abajo, se envía al monedero y, después de ser verificada, el saldo de éste se incrementa con la cantidad indicada. La información intercambiada durante este proceso de carga es la siguiente:

1. C → M: APDU(e-purse load initialitation)

Este comando simplemente refleja el envío al monedero del APDU que indica el comienzo del proceso de carga o incremento de saldo.

2. M → C: APD(S_M(SN_E,TN_E,RandNum,EPI),$Cert_{EP}$)

A continuación, el monedero firma la información de su estado actual que permitirá identificar al emisor que la petición proviene de un monedero y que, además, es una petición que se acaba de generar. En concreto, la información firmada contiene el número de serie (SN_{EP}), el número de transacción (TN_{EP}), el identificador del emisor de tarjetas inteligentes (EPI) y un número generado aleatoriamente (RandNum). Junto con la información firmada se envía en certificado del monedero ($Cert_{EP}$).

3. C → EPI: PM=(Payment(PI), $S_{EP}(SN_{EP}, TN_{EP}, H(PI), RandNum, EPI), Cert_{EP}$)

El cliente junto con la firma generada por la tarjeta envía la información de pago (PI) que utilizará para realizar la carga del monedero.

4a. EPI → C: $S_{EM}(SN_{EP}, TN_{EP}, RandNum, Amount, H(PM)), Cert_{EPI}$
4b. C → EP:
APDU(load e-purse:$S_{EPI}(SN_{EP}, TN_{EP}, RandNum, Amount, H(PM)), Cert_{EPI}$)
4c. EP → C: ADP(R_CODE,Balance)

donde:

- *Balance (Saldo)*. Indica el saldo disponible en la tarjeta después de realizar la operación de carga de monedero.

Una vez realizado el pago correctamente, el emisor de tarjetas genera una firma electrónica que actuará como moneda genérica que permite incrementar el saldo del monedero. Esta información se pasa al monedero y si el número de transacción, el número de serie del monedero y el número aleatorio coinciden con los que generó anteriormente en el paso 2, entonces el monedero incrementará su saldo en la cantidad indicada por el campo *Amount* (cantidad). Como resultado el monedero devuelve el saldo disponible en el monedero después de la carga (balance).

3.3.5. Moneda electrónica

En nuestra propuesta, una moneda electrónica tiene el siguiente formato:

$$e\text{-}coin_{EP} = S_{EP}(SN_{EP}, VnID, TID, RandNum, Date, Amount)$$

donde:

- *VnID* es un identificador de vendedor (habitualmente el resumen digital de su clave pública, aunque también podría ser otra información que pueda identificarlo, como podría ser su dirección de correo electrónico).

- *TID* es un identificador de transacción que proporciona el vendedor durante el proceso de pago y consiste en el resumen digital de la información de la transacción.
- *RandNum* es un número aleatorio que actúa como *nonce*.
- *Date* identifica la fecha en la que se realiza el pago.
- *Amount* indica el importe de la moneda.

3.3.6. Pago

El proceso de pago consiste en la generación de una moneda específica de vendedor que el usuario envía al vendedor por medio de un protocolo de pagos que hayan acordado entre ambos y que pueda garantizar el intercambio equitativo de este dinero electrónico como pago a un producto o servicio. Una vez que el vendedor haya recibido y verificado el pago, entregará o proporcionará el producto o servicio solicitado.

A continuación, en esta sección, explicaremos cómo se genera una moneda electrónica. Sin embargo, como se mencionó anteriormente, no especificaremos el protocolo para enviar la moneda y recibir el producto ya que no es parte de nuestra propuesta. Para este propósito podrían ser utilizados diversos protocolos como SPEED o los mencionados en [1, 144].

Para acuñar una moneda, el monedero necesita que el vendedor le facilite la información necesaria para acuñar una moneda específica para el pago de la transacción. Esta información aparece en el siguiente mensaje:

1. V → C: (VnID, TID, Date, Amount)

donde:

- *V* (*Vendedor*). Representa al vendedor.

A continuación, si el monedero dispone de saldo suficiente acuña la moneda y realiza un decremento de su saldo. Es importante destacar que podemos especificar la cantidad exacta de la moneda, es decir, la cantidad que se encuentra especificada en el monedero es divisible. Con el fin de autorizar la generación de la moneda, se solicita al usuario la introducción de su número PIN. Este número lo recibió el usuario cuando adquirió el monedero. La información intercambiada en este paso es:

2a. C → EP: APDU(generate e-coin: VnID, TID, Amount, PIN)
2b. EP → C:
APD(R_CODE,[S_{EP}(SN_{EP},VnID,TID,RandNum,Date,Amount),$Cert_{EP}$], Balance)
2c. C → V: S_M(SN_E,VnID,TID,RandNum,Date,Amount),CertEP

Cuando el vendedor recibe la moneda y el certificado comprueba que el certificado fue emitido por un emisor de monederos, que la moneda fue firmada con la clave privada asociada a este certificado y la cantidad de la moneda. En caso de que la información sea correctamente verificada, el vendedor procederá a proporcionar el producto o servicio solicitado.

3. V → C: Producto o servicio

Cabe destacar que en este proceso no hay necesidad de verificaciones on-line o de módulos SAM. No es necesaria una conexión on-line porque el vendedor tiene instalados todos los certificados de emisor y las CRLs (Certificate Revocation List – Lista de Certificados Revocados) que necesita. Aún así existe la posibilidad de comprobar el estado de un certificado mediante una conexión en línea con un mecanismo OCSP (Online Certificate Status Protocol – Protocolo en línea de comprobación del estado de un certificado). Posteriormente, en la sección de seguridad, analizaremos por qué no es necesario. A continuación, mostramos la información intercambiada durante el proceso de pago.

3.3.7. Depósito

El vendedor almacena todas las monedas que recibe como consecuencia de los pagos que los clientes realizaron por sus productos o servicios. Éstas serán enviadas, de forma conjunta, al final del día (o de cualquier otro período que se crea conveniente o se acuerde) al emisor. El conjunto de monedas será firmado por el vendedor. De esta manera, el vendedor recibirá el pago por las monedas recibidas. La información que envía el vendedor es la siguiente:

1. V → EPI: S_V(VnID,EPI,ID,TimeStamp,e-coin$_1$,…,e-coin$_n$)

donde:

- *ID* es un identificador de transacción que ha sido generado aleatoriamente.
- *TimeStamp (Sello de tiempo)* representa una cadena con la fecha y la hora a la que el vendedor envía la solicitud de depósito.
- *e-coin$_1$, …, e-coin$_n$* representa el conjunto de monedas que recibe el vendedor y cuyo importe debe pagar al vendedor.

Cuando el emisor recibe las monedas, primero verifica la firma del vendedor y, a continuación, pasa a verificar las monedas. Es decir, comprueba que la firma es válida y que tienen un identificador de monedero válido. A continuación, realiza el pago al vendedor. Normalmente, la cantidad que suman las monedas se envía a la cuenta bancaria del vendedor. Esta información la podría proporcionar el vendedor por medio de un

registro con el emisor que el vendedor puede realizar previamente o justo en el momento en el que recibe la primera moneda. Como resultado del proceso de depósito el emisor de monederos envía al vendedor:

2. EM → V : S_{EPI}(ID,VnID,Result,[Info])

donde:

- *Result* (*Resultado*) indica si la operación de depósito se realizó correctamente o no.
- *Info* (*Información*) representa información opcional que indica si la operación se realizó con éxito o no.

3.3.8. Análisis de las propiedades del monedero

En esta sección analizamos las distintas características que ofrece nuestra propuesta. En este análisis destacarán los distintos aspectos relacionados con la seguridad, mencionando en qué está basada la seguridad de los distintos elementos, los distintos procesos y como éstos soportan distintos requisitos de seguridad relacionados con los sistemas de pagos. En primer lugar presentaremos los resultados de la validación formal que llevamos a cabo y a continuación analizaremos en detalle las distintas propiedades de la propuesta.

Para este análisis consideraremos las mismas suposiciones que realizaremos en cuanto a la criptografía en la Sección 3.2.4, es decir, que las firmas son infalsificables, que las funciones de resumen son libres de colisiones y que existe una autoridad de certificación en la que podemos confiar. Además, supondremos que la tarjeta inteligente, que implementará el monedero aquí propuesto como un applet Java Card, es segura frente a diversos tipos de ataques como análisis del canal, análisis del tiempo de ejecución o del consumo (diferencial) de energía del dispositivo, etc. (veáse Sección 2.2.3).

El applet de la tarjeta gestiona la clave privada y puede llevar a cabo las operaciones descritas en la Sección 3.3.2. Este applet impedirá que la clave privada salga de la tarjeta inteligente. Asimismo el applet, gracias al mecanismo de cortafuegos que proporcionan las tarjetas Java Card, estará protegido de los ataques de otros applets que quieran acceder a esta información.

Validación formal

En este apartado comentamos la validación formal hemos realizado de los protocolos que intervienen en las distintas fases en las que se utilizar el monedero. Esta validación está basada en el uso de la herramienta AVISPA (Automated Validation of Internet Security Protocols and Applications) [9, 11, 303]. Las principales características de esta herramienta y el proceso a seguir ya fueron anteriormente presentadas en la Sección 3.2.4 cuando se presentó la validación formal del protocolo SPEED.

Siguiendo dicho proceso, se especificaron los protocolos de certificación y de carga del monedero en HLPSL. Del proceso de pago no se llevó a cabo porque depende del protocolo

elegido para llevar a cabo el intercambio (si se considera necesario). A continuación, dichas especificaciones han sido analizadas con los distintos módulos que proporciona AVISPA y podemos afirmar que la herramienta no ha encontrado ningún ataque. Estas especificaciones formales en HLPSL se pueden encontrar en los Apéndices D y E respectivamente.

Generación de dinero falso

Uno de los principales problemas a los que se tiene que enfrentar un sistema que genera monedas electrónicas es la generación de dinero sin que al final el usuario tenga que pagar por él, es decir, la generación falsa de dinero.

Para poder generar dinero sin tener que pagar por él, un atacante podría intentar los siguientes ataques:

- Obtener la clave privada del monedero.
- Certificar una clave privada que no se encuentre protegida por el monedero.
- Incrementar el contador de dinero disponible en el applet.
- Suplantar al emisor en el proceso de carga del monedero.

A continuación, en las siguientes secciones se explicarán los distintos mecanismos de seguridad que se ofrecen para proteger al monedero de estos ataques. Posteriormente, se analizarán otras características como el doble gasto, el intercambio equitativo, anonimato, etc.

Seguridad de la clave privada del monedero

Para generar dinero en esta propuesta se dispone de una clave que está certificada por un emisor de tarjetas. Un atacante si quisiera generar dinero sin el monedero tendría que conseguir firmar cierta información con la clave mencionada. De esta forma, el control del uso de esta clave estaría en poder del usuario en lugar del monedero y éste podría generar dinero sin que haber tenido que pagar previamente por él. Este ataque podría ser exitoso por dos vías que explicamos a continuación.

La primera posibilidad sería conseguir la clave privada utilizada para firmar las monedas. Sin embargo, este ataque no es posible salvo que se obtenga la clave del monedero, caso que no es posible como explicamos más abajo, o salvo que se comprometa la clave privada. Estamos suponiendo que el algoritmo de firma utilizado y la longitud de clave son lo suficientemente seguros para no ser falsificados. Por tanto, comprometer una clave privada no sería posible. La segunda posibilidad sería conseguir suplantar a un monedero en el proceso de certificación. Este ataque será analizado en la siguiente sección.

Como decíamos, una de las posibilidades de un atacante es obtener la clave privada del monedero. La clave privada se genera dentro del applet de monedero de la tarjeta. Este applet impedirá el acceso a esta clave desde el exterior por los mecanismos de protección que ofrece la tarjeta inteligente (ver Sección 2.2.3). Este applet, además, está protegido de ataques internos producidos por otros applets residentes en la tarjeta por el mecanismo de

cortafuegos. Como se menciona en [187, 234], estos dispositivos podrían ser considerados resistentes a ataques de manipulación y, por tanto, el coste asociado a obtener dicha información sería superior al beneficio obtenido.

Además, hay que considerar, que una vez detectada la existencia de monedas falsas (el emisor puede controlar el dinero máximo que tiene disponible un determinado certificado y cuánto ha gastado), el certificado de monedero será revocado y, por tanto, la clave privada asociada. El certificado revocado se pone en una lista de certificados revocados que será distribuida a los vendedores interesados.

Al considerar el monedero un dispositivo resistente a ataques de manipulación donde el coste de romper el sistema es mayor que el beneficio, se puede asumir el riesgo de no comprobar el estado del certificado mediante mecanismos on-line como OCSP y, por tanto, no hay necesidad de una conexión en línea para verificar una moneda electrónica; sería suficiente con que cada cierto intervalo de tiempo se instalaran las correspondientes CRLs. Este es el mismo riesgo que se asume en otras propuestas como CEPS o EMV. Aunque dependiendo del escenario se podría considerar el uso de mecanismos on-line.

De esta forma podemos estar seguros de que nadie utilizará la clave de firma de generación de monedas sin la tarjeta, evitando así que se puedan generar monedas por las que el usuario no pagará.

Además, para garantizar que las claves utilizadas para firmar estas monedas corresponden a un monedero, se ha definido un proceso de certificación. La seguridad de este proceso se analiza en la siguiente sección.

Seguridad del proceso de certificación

Para conseguir certificar una clave privada que no esté bajo el control del monedero sería necesario suplantar a éste en el proceso de certificación. Otra posibilidad sería obtener/comprometer la clave privada del emisor de monederos. Sin embargo, esa situación no puede darse bajo los supuestos de seguridad planteados.

En el proceso de certificación, que ha sido formalmente con AVISPA, el monedero obtiene un certificado en base a una clave diversificada que instaló el emisor en el monedero. Esta clave nunca se intercambia entre el monedero y el módulo SAM para evitar que la clave pueda ser interceptada. Esta clave se utiliza en el proceso de certificación para “firmar” la solicitud de certificación y un reto. Gracias al uso de esta clave, el emisor puede estar seguro de que la petición de certificación viene de un monedero que satisface los requisitos establecidos en la Sección 3.3.2 ya que el applet fue personalizado por él mismo. Además, a esta clave tampoco sería posible acceder gracias a los mecanismos de seguridad definidos tanto por la tarjeta como por el applet. Este mecanismo de seguridad es similar al que define el estándar de monedero WG10 para sus transacciones (ver Sección 2.2.4). Por tanto, bajo las condiciones que acabamos de describir, esta segunda vía de ataque tampoco es posible.

Seguridad del proceso de carga del monedero

Para poder incrementar el saldo del monedero sin tener que realizar un proceso de pago a la entidad emisora existen dos posibilidades. La primera sería atacar al applet para modificar el contador del saldo. Este ataque no es posible por los mecanismos de protección del applet y la tarjeta descritos anteriormente. La otra posibilidad sería intentar suplantar al emisor de tarjetas en el proceso de carga del monedero. La seguridad de este proceso la analizamos a continuación.

En el proceso de carga, la seguridad de la transacción completa depende de varios elementos. Primero, del protocolo utilizado para pagar y recibir las monedas. Segundo, de la propia seguridad de la moneda generada por el emisor para cargar el monedero. Finalmente, de cómo se realiza el incremento del saldo del monedero.

El protocolo utilizado para pagar y recibir la moneda que permitirá incrementar el saldo del monedero debe tratarse de un protocolo que garantice un intercambio equitativo de bienes. Para este propósito se podría utilizar el protocolo de pago SPEED presentado en la Sección 3.2 de esta tesis o cualquier otro que se considere adecuado para este fin. En el caso específico de SPEED, el producto que el cliente compraría sería la moneda que le permitiría incrementar el saldo de su monedero. En cuanto al pago, en este caso el cliente utilizaría su tarjeta de crédito o un número de cuenta.

La seguridad de la moneda que el cliente recibe del emisor depende de que alguien pueda falsificar una firma del emisor. Bajo los supuestos de seguridad planteados esta situación no puede darse. Por otro lado, si la clave privada del emisor fuera comprometida, el certificado del emisor sería revocado y, por tanto, los vendedores ya no aceptarían más pagos de los monederos certificados con tal clave. Con el fin de evitar que el usuario pueda pagar a un emisor de tarjetas con la clave comprometida, es importante que previamente compruebe la validez del certificado. Por tanto, bajo estos supuestos, el monedero sólo incrementa su saldo si recibe una moneda firmada por el emisor incorporando la información indicada en el paso 2 del proceso de carga del monedero (ver Sección 3.3.4). Así, evitamos que el cliente pueda generar su dinero electrónico.

Otro posible ataque consistiría en enviar varias veces al monedero la misma moneda que se recibió del emisor para así incrementar el saldo varias veces a partir de un solo pago. Sin embargo, este ataque de reenvío no es posible, ya que cada vez que el proceso de carga se inicia, el monedero genera un número aleatorio e incrementa el contador de transacciones. Así, para que una moneda recibida pudiese ser utilizada más de una vez para incrementar el saldo, debería tener estos valores así como el valor resumen digital de la información de pago asociada al proceso de carga (véase pasos 3 y 4 del proceso de carga). Este número junto con el resumen digital actúan como una etiqueta única tal y como se propone en [1, 120]. Por tanto, a menos que el generador de número aleatorios no sea lo suficientemente seguro y, por tanto, la información de pago sea siempre la misma, este ataque no será posible.

Doble Gasto

Un importante aspecto en cuanto a la seguridad de cualquier sistema de pago es el conocido doble gasto. Es decir, que un usuario o un vendedor puedan realizar distintos pagos con la misma moneda. Un usuario podría intentar utilizar una moneda con distintos vendedores o incluso con el mismo.

Aunque el cliente intente utilizar la misma moneda con distintos vendedores, el resultado no va a ser exitoso ya que las monedas están personalizadas para cada vendedor mediante un identificador de vendedor único que, como se comentaba, normalmente es la clave pública del vendedor. De esta manera, el vendedor puede comprobar si la moneda fue generada para él o no.

Otro intento de doble gasto que podría acometer el usuario sería el envío de la misma moneda al mismo vendedor. Sin embargo, esta situación también puede ser detectada por el vendedor ya que en el proceso de pago, el vendedor proporciona al cliente un identificador único de la transacción (el valor *TID*) que debe incluir en la moneda. Si este identificador no está incluido en la moneda, el vendedor la rechazará. De esta forma el vendedor se asegura de que es una moneda generada para esa transacción y que no fue utilizada previamente.

Finalmente, una tercera parte distinta de los involucrados en la transacción podría intentar copiar las monedas enviadas por el cliente al vendedor durante su envío en la fase de pago (en el caso de no utilizar un protocolo no seguro). Pero aunque las monedas fueran copiadas, sólo el vendedor especificado podrá depositarlas ya que son específicas para éste. En la moneda se ha incluido el resumen digital de la clave pública del vendedor o cualquier otra información que, posteriormente, en el proceso de depósito pueda acreditarlo de forma unívoca.

Intercambio equitativo y atomicidad

En el proceso de pago, aparte de tener en cuenta el doble gasto, también hay que garantizar otro aspecto como es el intercambio seguro (equitativo) de la moneda a cambio del producto o servicio solicitado. En este sentido, esta propuesta está abierta y no establece ningún protocolo, dando libertad a la elección de diversos protocolos dependiendo de los importes a pagar o del valor del producto a adquirir en un determinado escenario.

Excepto en el caso de los sistemas de micropagos, en el resto de sistemas, en la fase de pago se debería utilizar un protocolo que garantice el intercambio equitativo de valores y el no repudio, proporcionado evidencias de la participación de las partes en la transacción. Uno de esos protocolos podría ser SPEED, descrito en la sección 3.2 de este capítulo, así como otros mencionados en [120, 163]. Por tanto, la seguridad de este intercambio dependerá del protocolo elegido para tal fin.

Independientemente del protocolo elegido, la moneda electrónica generada proporciona evidencias de la participación del monedero electrónico en la transacción. Por un lado, la moneda contiene información relacionada con la transacción (*TID*) y el identificador del vendedor (*VnID*). Por otro lado, la firma de la moneda sólo puede ser generada por un

monedero. Así, el vendedor puede estar seguro de que una vez que ha recibido la moneda, posteriormente cobrará ya que, a menos que las claves del monedero hayan sido comprometidas, es casi imposible acuñar una moneda. Por tanto, cualquier moneda que el vendedor reciba será aceptada como pago correcto, siempre y cuando ésta no haya sido entregada previamente.

Privacidad y Anonimato

Las monedas generadas no contienen ninguna información personal relacionada con el usuario, por lo tanto, en principio, el pago es anónimo. Sin embargo, este anonimato se vería comprometido y sería trazable si el usuario revelara alguna información personal al vendedor.

En el momento en que el vendedor obtenga información personal del usuario, puede asociar todas las transacciones que tienen el mismo identificador de monedero a ese usuario. A partir de ese momento el vendedor podría colaborar con otros vendedores o con el emisor para intercambiar información y construir el perfil del usuario.

Para el emisor, la información es anónima siempre que se den dos condiciones. Primera, que el proceso de distribución de las tarjetas sea anónimo (por ejemplo, mediante su compra en un quiosco de forma similar a como anteriormente se hacía para las tarjetas de prepago). Segunda, que en el proceso de carga del monedero el emisor no obtenga información personal del usuario. En el momento en que no se satisfaga alguna de estas condiciones, el usuario podría ser rastreado y el emisor podría conocer los distintos vendedores con los que ha colaborado. Sin embargo, para obtener un perfil sería necesario que colaborara con el vendedor para obtener información acerca de los productos o servicios consumidos. En este sentido, el anonimato de nuestra propuesta es similar a otros monederos como WG10 o CEPS (ver sección 2.2.4).

Finalmente, con respecto a la privacidad de la información intercambiada durante el proceso de pago, mencionar que ésta dependerá del protocolo de pago realizado para realizar el intercambio de los bienes por la moneda electrónica. Si el protocolo garantiza la confidencialidad, nadie excepto los participantes podrán obtener información acerca del producto vendido/comprado.

Divisibilidad

El monedero PURSE-COIN se carga por medio de una moneda emitida por el emisor de tarjetas. Este valor se registra en un contador. Posteriormente, el cliente puede realizar un pago por cualquier valor que sea inferior a esta cantidad. Por tanto, podemos afirmar que el esquema aquí propuesto satisface la propiedad de divisibilidad.

3.3.9. Conclusiones

Las tarjetas inteligentes presentan interesantes características de seguridad y portabilidad, por lo que parece interesante utilizarlas como método de pago. Sin embargo, tal y como analizábamos en el capítulo anterior, existían dos problemas que han llevado a que su uso no sea tan extendido. El primer problema era la escasa difusión de las tarjetas

inteligentes, problema que a día de hoy se ha mitigado debido a EMV y al DNI electrónico. El segundo y principal problema de las propuestas aparecidas hasta ahora es el rechazo a su uso para pagos por Internet debido al hecho de que requieren que los vendedores incorporen el uso de dispositivos hardware tipo SAM que pueden llevar a que el proceso de pago sea lento. Una primera aproximación para solucionar este último problema era SPEED. Aún así, SPEED no elimina la necesidad de la interconexión con un módulo SAM.

En respuesta a esta problemática, hemos presentado un monedero electrónico denominado PURSE-COIN. Este monedero está basado en la generación de monedas específicas de vendedor lo cuál facilita tanto el proceso de pago, ya que no es necesario el intercambio de mensajes entre dos dispositivos hardware remotos, como la verificación sin el uso de una tercera parte y un mejor control del doble gasto. Además, el monedero ha sido diseñado de forma segura para evitar problemas de doble gasto, generación de monedas falsas, etc. Estas propiedades de seguridad han sido analizadas exhaustivamente. Como parte de este análisis se ha incorporado una validación formal basada en la herramienta de validación automática AVISPA [9, 11, 303].

Esta propuesta, al estar basada en el uso de una aplicación que puede ser instalada en cualquier tarjeta Java Card, tiene la ventaja adicional de que podría ser ampliamente extendida debido a que, a día de hoy, casi todo el mundo posee una tarjeta inteligente en su teléfono móvil. Por tanto, esta solución tiene unas amplias posibilidades de extensión al conocido como comercio móvil, sobre todo teniendo en cuenta que con la nueva generación de teléfonos móviles las velocidades de transmisión de datos se verán incrementadas y permitirán enviar fácilmente estas monedas electrónicas. Otro hecho que apoya su uso en dispositivos móviles es la aparición de tecnologías (como SATSA – Security and Trust Services API for J2ME [284]) que permiten, desde el dispositivo móvil, comunicarse con la tarjeta inteligente para generar firmas, monedas, etc.

Finalmente, el uso de este sistema de pagos podría utilizarse en combinación como protocolos que garanticen el intercambio equitativo de bienes (bien on-line, como SPEED, u off-line como [97, 305, 306]). En el caso del uso de un protocolo off-line, como resultado obtendríamos un protocolo seguro para la distribución de contenidos electrónicos en B2C y que, además, sería más eficiente que su combinación con SPEED puesto que no necesitan la participación de una TTP on-line. Hay que tener en cuenta que SPEED está especialmente pensado para aquellos vendedores que quieren soportar algunos de los esquemas de monedero ya existentes y que requieren el uso de una tercera parte que contenga los dispositivos hardware que permitan realizar el proceso de pago. A la hora de decidirse por una opción u otra también habría que tener en cuenta si se desea realizar la negociación del precio del producto. SPEED si incorpora esta característica, sin embargo, las soluciones off-line que hemos mencionado no la incorporan. En las soluciones off-line entonces sería necesario que extendieran el protocolo para soportar esta característica o tendrían que combinar el protocolo con un protocolo de negociación.

3.4. Conclusiones del capítulo

En los últimos años el uso de las nuevas tecnologías y la creación de contenidos en la Web han obtenido un gran auge. Una de las principales áreas que se ha beneficiado de este auge ha sido el comercio electrónico, en particular el comercio B2C. En él, los distintos creadores, editores, productores y distribuidores de contenido electrónico quieren obtener beneficios de los contenidos que crean, editan, producen y distribuyen respectivamente. Para este propósito están los mecanismos de pago.

Como presentamos en el capítulo anterior, muchas de las soluciones propuestas no securizaban todo el proceso de compra incluyendo negociación, pago y distribución. También presentamos las tarjetas inteligentes y vimos como éstas pueden aportar un mayor grado de seguridad en la protección de nuestra información criptográfica y de dinero electrónico. Además facilitan la movilidad del usuario y el uso de ese dinero electrónico en distintos escenarios.

Debido a las ventajas que presentan los monederos electrónicos, decidimos ofrecer soluciones de pago B2C para la adquisición de productos electrónicos basados en éstos. Con este objetivo surgían dos retos principales: el primero solucionar los problemas ya planteados con los sistemas de pago. El segundo reto era resolver el problema que presentan los monederos electrónicos. Es decir, evitar la necesidad de integrar en los servidores de pago de los vendedores un dispositivo hardware, lo que no los hace muy adecuadas para pagos en Internet.

Estos problemas los hemos intentado solucionar de una forma gradual. En primer lugar, hemos presentado un protocolo de pagos denominado SPEED. Este protocolo, resuelve los problemas de los sistemas de pago para la adquisición de bienes electrónicos ofreciendo seguridad en todas las fases de la compra y garantizando la buena atomicidad. Además, facilita el uso de monederos electrónicos ya existentes como WG10, CEPS o EMV a los vendedores. Esta solución se ha probado en diversos escenarios y además se ha comprobado que ofrece una buena escalabilidad. Sin embargo, aunque se resuelve el problema para el vendedor, éste se translada a otra entidad, el broker. Éste es el encargado de velar por la seguridad del sistema e integrar los dispositivos hardware.

Con el fin de facilitar aún más el uso de los monederos electrónicos en las transacciones en Internet realizamos una nueva propuesta de monedero denominada PURSE-COIN. Esta propuesta combina el uso de los monederos electrónicos con la generación de monedas electrónicas. Si esta propuesta, además, la combinamos con un protocolo que garantice el intercambio equitativo y que esté basado en una TTP off-line, obtendremos un protocolo seguro y eficiente para distribución de contenidos en entornos B2C. Si, además, PURSE-COIN la unimos al hecho de que podría introducirse en las tarjetas SIM de nuestro teléfonos móviles, en este caso estaríamos hablando de un sistema de pago que podría ser utilizando tanto en entornos de comercio electrónico como en entornos de comercio móvil, así como en máquinas de vending. Por tanto, estaríamos ofreciendo una solución válida para múltiples escenarios de pago.

Es importante comentar que además de intentar facilitar el uso de las tarjetas hemos tenido en cuenta la seguridad de nuestras propuestas, ya que como decíamos, uno de los

principales aspectos de cualquier sistema de pagos es la seguridad que éste ofrece. Ambas propuestas han sido analizadas formalmente con una herramienta de validación automática y no se ha encontrado ningún ataque.

Por tanto, en este capítulo podemos afirmar que hemos aportado dos soluciones de pago basadas en tarjeta inteligente que pueden cubrir un amplio espectro de soluciones de pago y que podrían ser demandadas una vez que el uso de las tarjetas sea familiar para los usuarios. A esta extensión y familiarización de las tarjetas van ha contribuir dos hechos fundamentales. En primer lugar, a la aparición de la directiva europea de firma electrónica que equipara la firma electrónica y la firma manuscrita, dotando a la primera del mismo marco legal que a la segunda. Esta directiva exige el uso de un dispositivo seguro de creación de firma, entre los que se encuentran principalmente las tarjetas inteligentes. Segundo, la extensión del uso de las tarjetas inteligentes gracias a la introducción del DNI electrónico en países como España o Finlandia que, además, satisface la especificación de dispositivo seguro de creación de firma.

Sin embargo, en los requisitos que se deberían satisfacer para obtener la confianza en los sistemas de pagos, la seguridad no era el último aspecto. También se planteaba la necesidad de uniformizar el modo en el que los usuarios utilizan cada uno de los distintos sistemas de pagos.

A lo largo de este Capítulo 2 presentamos distintos protocolos de pagos, además, en este capítulo se han introducido dos nuevas propuestas y, en el futuro, aparecerán nuevos protocolos que mejorarán éstos o que serán más adecuados para un determinado entorno particular u ofrecerán determinadas características que los harán de más valor frente a éstos en determinados escenarios. Por tanto, de cara a obtener la confianza de los usuarios en los sistemas de pagos, aparte de la seguridad, tenemos que ofrecer un marco que permita utilizar distintos mecanismos de pago pero que, a la misma vez, el modo de efectuar este pago sea uniforme para los usuarios, de forma que estos estarán seguros de que realizan el proceso correctamente. Además, este marco es necesario de cara a facilitar que los distintos vendedores puedan ofrecer distintos mecanismos de pagos, así como incorporar nuevos mecanismos que posteriormente puedan aparecer. Estos aspectos serán abordados en el siguiente capítulo.

Capítulo 4

Frameworks de pago para contenidos y servicios Web

La proliferación de sistemas de pago es cada vez mayor y, paulatinamente, las propuestas ofrecidas son más seguras y ofrecen más flexibilidad intentando cubrir distintos tipos de escenarios y necesidades, tal y como hemos podido analizar a lo largo del Capítulo 2 así como en el Capítulo 3 con las mejoras presentadas.

En el comercio B2C, la principal utilidad de estos sistemas de pago se encuentra en el ámbito de la compra de contenidos y servicios electrónicos. A la hora de realizar el proceso de pago, uno de los principales inconvenientes que se encuentran los usuarios, vendedores y desarrolladores de este tipo de sistemas es realizar el pago cuando existen distintas opciones. Éstos se encuentran que no existen interfaces bien definidas, sencillas, que les resulten familiares y que les generen confianza al realizar este proceso. Para este propósito, como veíamos en el Capítulo 2, se proponían los frameworks de pago. Sin embargo, tal como analizamos en ese mismo capítulo, no existe una solución genérica que contemple todos los aspectos necesarios para la adquisición de productos y/o servicios en la Web.

En este capítulo abordaremos cómo definir frameworks de pago que faciliten el pago de contenidos y/o servicios Web. De esta forma, el usuario podrá elegir entre los protocolos presentados en el Capítulo 2 (PayCash, Payword, SET, etc) o en los nuevos presentados en el Capítulo 3 (SPEED, PURSE-COIN) a la hora de realizar sus compras en la Web.

Nuestro framework de pagos B2C para contenidos Web busca que el proceso de pago se pueda realizar de forma muy sencilla, con sólo realizar un clic en un enlace. El framework, por tanto, soporta distintas opciones de pago así como una serie de características asociadas. Entre estas características destacan que define la información a incorporar en la Web indicando distintos precios por las distintas opciones, permite la negociación del precio, ofrece un protocolo genérico de intercambio de mensajes que garantiza la seguridad de la información intercambiada y es extensible. El framework, además ha sido implementado y evaluado. Los resultados obtenidos se presentarán a lo largo de este capítulo.

Los distintos componentes que forman parte del framework ha sido definidos de la forma más genérica posible con el fin de que éstos se puedan utilizar en otros escenarios y con las posibles futuras propuestas que aparezcan. En particular estos componentes se utilizarán para la definición de un framework para el pago de servicios Web que también presentamos en este capítulo.

4.1. Introducción

Los pagos electrónicos son uno de los elementos fundamentales dentro del comercio B2C y, a día de hoy, resultan de gran valor para el pago de contenidos electrónicos que se distribuyen a través de Internet, tal y como analizamos a lo largo del Capítulo 2.

Su uso es particularmente interesante en dos escenarios relacionados con la Web. En primer lugar, a la hora de pagar por los enlaces visitados de una página Web [67, 117, 132, 192], por ejemplo, para el pago de artículos de un periódico, revista o conferencia, ficheros de música, etc. Este modelo es conocido como *pagos por clic* (*per-fee-links*) y ha sido propuesto como una alternativa tanto a los anuncios en la Web como a las suscripciones, como se puede ver en Peppercoin [237], W3C per-fee-links [193], en [132], etc. En segundo lugar, para realizar el pago por el uso de servicios Web donde un cliente solicita a un determinado proveedor de servicios que le suministre información, realice una determinada tarea, etc.

En ambos escenarios se debe facilitar al usuario la posibilidad de elegir el mecanismo de pago que soporte o considere más adecuado. Sin embargo, a día de hoy, no existen frameworks de pagos genéricos para realizar el pago por productos o servicios Web que los vendedores puedan seguir y que garanticen o que provoquen la confianza de los usuarios, de forma que éstos realicen pagos en la red de una forma segura y confiable. Como Gefen et. al. argumentan en [118], las bases para sentar la confianza de los usuarios en los pagos on-line pasan por:

a) una confianza en que el vendedor no tiene nada que ganar engañando,

b) una confianza en que hay mecanismos de pago seguros para incorporar en la Web,

c) tener una interfaz característica, y

d) que sea fácil de usar.

Sin embargo, hasta el momento, cada método de pago usa su propio sistema propietario. Este hecho provoca desconfianza en los usuarios ya que ven que la forma de pagar con cada vendedor y con cada sistema de pago es diferente y, por tanto, no están seguros de si el procedimiento seguido es el adecuado.

Otra consecuencia que se deriva del uso de sistemas propietarios es que el software de diferentes vendedores no interopera y, por tanto, la introducción de sistemas de pago en la Web se hace compleja y lenta. De hecho, ésta es una de las razones por las que, a pesar de que existen un amplio número de sistemas de pago seguro, al final, los pagos se llevan a cabo por medio de tarjetas de crédito usando una conexión SSL/TLS [73, 236], ya que es uno de los métodos más cómodos para los usuarios y de los más valorados [266].

Como respuesta a esta problemática, en el Capítulo 2 veíamos que los frameworks de pagos aparecerían para facilitar el uso de distintos sistemas de pago en las aplicaciones. Sin embargo, veíamos que las propuestas realizadas hasta ahora no son adecuadas o no cubren todos los elementos necesarios para realizar pagos por contenidos o servicios Web.

Las características que un framework de este tipo debe soportar fueron descritas en la Sección 2.3.1, a continuación, las mostramos de forma resumida:

- Debe facilitar la negociación y elección de las *opciones de pago*. Es más, como característica de valor añadido sería deseable que soportara la negociación del precio del producto a la misma vez que negocia las opciones de pago.
- El framework debería poder encapsular los diferentes mensajes del protocolo elegido para efectuar el pago y así poder recibir el contenido pagado. Por este motivo, el framework debería proporcionar un protocolo independiente del sistema de pagos concreto a utilizar, de forma que se soporte cualquiera de los protocolos de pagos existentes hasta la fecha, así como futuras propuestas de pago que aparezcan. Es más, este protocolo genérico debería ser eficiente a la hora de enviar pagos de bajo valor o micropagos, así como soportar la realización de pagos de forma repetida o consecutiva con el mismo vendedor de una forma eficiente [81]. También debe contemplar otros posibles modelos de negocio.
- El método para la incorporación de un nuevo mecanismo de pago debería ser por medio de una interfaz de programación de aplicaciones (API – Application Programming Interface) para los distintos wallets que soporten el framework. Esta API debería ser genérica y ocultar las particularidades de cada protocolo.

Para el escenario de pagos por clic además se debe satisfacer la siguiente propiedad:

- Se debe distinguir claramente entre aquellos enlaces que son de pago y aquellos que no lo son, además de adjuntar la información relacionada con el pago en el correspondiente enlace. Así, el cliente puede conocer la información relacionada con el pago sin tener que iniciar el proceso de pago. El hecho de definir, de una manera uniforme, la información de pago facilita su procesamiento automático así como su descubrimiento y el proceso de elección de pago. De esta manera, se podría automatizar el proceso de elección de pago por medio de agentes inteligentes o cualquier otro mecanismo de procesamiento automático. Esta información, además, debería ser definida de la forma más genérica posible, de forma que se pueda utilizar en futuros sistemas que puedan aparecer o en otros escenarios como, por ejemplo, para la negociación de los productos y sus precios o el pago de servicios Web.

En este capítulo proponemos dos frameworks de pago para productos y servicios Web que satisfacen estas características. Para su definición hemos tenido en cuenta las principales aportaciones realizadas por las propuestas de frameworks presentadas en el Capítulo 2 y hemos cubierto las deficiencias que encontramos en éstas. Para conseguir estos objetivos, en primer lugar, presentaremos el framework de pagos por clic, donde se definen los elementos genéricos que deben formar parte de cualquier framework, así como

los elementos específicos para este modelo. A continuación, basándonos en los elementos definidos en este framework, desarrollaremos el framework para el pago de servicios Web.

4.2. Framework de Pagos por Clic

El modelo de pagos por clic es interesante ya que permitiría a los vendedores no tener que insertar publicidad en sus páginas para obtener beneficios, lo cual hace más atractivo el sitio Web a los clientes. En este caso, el beneficio vendría de los contenidos que directamente vende a un precio muy bajo, lo que conlleva atraer a un gran número de clientes y así, a pesar de que los importes son pequeños, obtener un volumen de ventas elevado que permita obtener los beneficios esperados. Por otro lado, los clientes son reticentes a las suscripciones y prefieren pagar sólo por los contenidos que realmente desean, por lo que esta alternativa también les resultaría atractiva.

En este modelo, como se explica en la iniciativa del W3C [193], se debe facilitar al usuario la posibilidad de elegir el mecanismo de pago que soporte o considere más adecuado. Sin embargo, como analizábamos en el Capítulo 2, a día de hoy, no existe un framework de pagos genérico para soportar este modelo.

En respuesta a esta problemática, en esta sección presentamos un framework de pagos genéricos que cumple todos los requisitos definidos en la introducción y que ha sido desarrollado y evaluado.

A continuación explicaremos en detalle cómo es el proceso de pago, cuáles son los distintos elementos que lo componen y los resultados obtenidos a partir de la implementación desarrollada.

4.2.1. Visión Genérica del Framework de Pagos por Clic

En esta sección, introducimos los componentes que forman parte de nuestro framework de pagos, así como también describimos el proceso que se seguiría para realizar el pago de un producto electrónico que está disponible a partir de una página Web.

En nuestro framework [253], los actores principales son: un cliente, que hace uso de su navegador y el servidor Web del vendedor. Adicionalmente, dependiendo del protocolo de pago a utilizar para realizar la compra, podría participar un *proveedor de servicios de pago* (PSP – entidad que acepta pagos en representación del vendedor), broker o intermediario. En general, en el caso de los micropagos, esta entidad no participa en la fase de pago ya que el principal objetivo de estos esquemas de pago es la eficiencia.

En la Figura 4.1 se muestra el proceso genérico que se sigue en el framework a la hora de adquirir un producto electrónico. Supongamos que un usuario quiere adquirir un contenido electrónico, como podría ser un artículo (en HTML o PDF, etc.), un archivo MP3, una imagen, etc. Después de navegar por la red, el usuario descubre un vendedor que ofrece tal contenido y efectúa un clic sobre la URL del vendedor en el navegador (paso 1). A continuación, el navegador enviaría una solicitud al servidor Web (paso 2) para acceder a la página del vendedor con el fin de obtener ese contenido (paso 3). Esta página, que se mostraría en el navegador (paso 4), podría contener uno o más enlaces de

pago que ofrezcan el contenido. Si el cliente está dispuesto a pagar por el contenido, realizaría un clic sobre el enlace de pago (paso 5). A continuación, el navegador y el servidor establecerían una sesión para negociar las opciones de pago y realizar la compra. Así, se negocian los protocolos de pago, las marcas financieras y los PSPs (*opciones de pago*) a usar (paso 6). El servidor, para las distintas opciones de pago, indicaría los precios del contenido al usar cada opción concreta.

Es importante mencionar que para un determinado producto los precios pueden variar dependiendo de la opción de pago elegida. La razón de permitir especificar distintos precios se debe a que el coste de todos los protocolos no es siempre el mismo, ya que éste puede variar dependiendo del coste computacional asociado a los mensajes, de las comisiones que aplique la entidad financiera, etc. El protocolo también permite la negociación del precio asociado a las distintas opciones de pago a la misma vez que éstas se negocian.

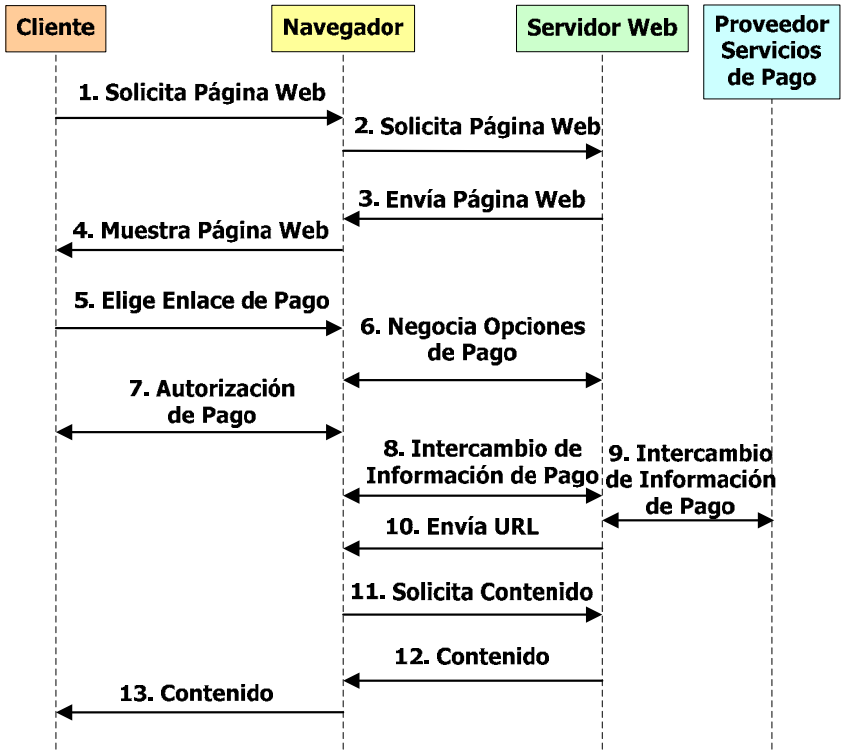


Figura 4.1. Proceso genérico de pago en el framework de pagos por clic.

Una vez que las opciones de pago han sido intercambiadas, el navegador podría interrogar al usuario acerca del mecanismo de pago a utilizar, si es que hay más de uno en común. El que se realice este proceso de consulta al usuario depende de la configuración, ya que el proceso de elección lo podría llevar a cabo por sí sólo el navegador de forma

automática o en función de una configuración del usuario previamente establecida. Además, el navegador solicita la autorización del usuario para iniciar el proceso de pago (paso 7). Esta confirmación podría ser simplemente una ventana de aceptación o podría requerir algún tipo de autenticación; por ejemplo, basada en PIN o en contraseña. Este paso es opcional dependiendo de la configuración elegida por el usuario, por ejemplo, en micropagos el usuario podría decidir que sólo se le pida su autorización cada X euros o cada vez que se envíen X micromonedas, etc.

Si el usuario acepta el pago, el protocolo de pagos se ejecuta y se intercambian los mensajes necesarios para efectuar la compra (paso 8). Dependiendo del protocolo elegido, podría tener que participar un PSP (en ese caso se ejecutaría el paso 9). En general, es el vendedor quien está conectado a esta entidad. Sin embargo, existen protocolos de pago donde el usuario es quien envía la información al PSP, como por ejemplo en SPEED [249], o en los modelos de pago basado en recibos [171].

Una vez que el pago se ha efectuado, el servidor envía la URL (junto con la información que pueda ser necesaria para su acceso como podría ser una cookie o algún token de autenticación) que realmente permitirá al navegador acceder al contenido solicitado por el usuario (paso 10). El navegador utiliza esta nueva URL para acceder al contenido que se acaba de comprar (pasos 11). De esta manera, el servidor proporciona el contenido (paso 12). Finalmente, el contenido se muestra al usuario (paso 13). Dependiendo del contenido, se podría lanzar una aplicación externa como un reproductor de MP3 o un visor de documentos PDF.

Cuando el usuario realiza un pago al vendedor, este pago podría proporcionar acceso a un único contenido o podría, por el contrario, dar acceso a varios contenidos, todo dependerá del modelo de negocio. Si el pago da derecho a acceder a varios contenidos, en las siguientes solicitudes de enlaces de pago el servidor Web proporcionará una URL para acceder al contenido sin solicitar un pago. Esta es la razón por la que hemos decidido realizar un framework orientado a sesión que sea capaz de mantener una sesión entre el navegador y el servidor. Así, el servidor puede controlar más fácilmente si el usuario ha realizado o no previamente un pago. Además, el servidor puede utilizar esta sesión para solicitar en cualquier momento un nuevo pago. El modo en el que se soporta cada uno de estos modelos será explicado posteriormente en la Sección 4.2.3.

Los componentes que se necesitan para soportar este framework de pagos por clic son los que se mencionarán a continuación. Posteriormente, cada uno de ellos será descrito en las siguientes secciones de este capítulo. Finalmente, se comentarán los detalles de la implementación realizada del framework que nos ha permitido probar y evaluar la solución propuesta. Los componentes que definimos son:

1. Para soportar el framework es necesario incorporar una serie de elementos tanto a la arquitectura del navegador como del servidor Web. El objetivo de estos elementos a añadir es soportar distintos protocolos de pago, el intercambio de los mensajes de pago de estos protocolos así como reconocer los enlaces de pago. Estos componentes se describen más en detalle en la Sección 4.2.2.
2. Como parte importante de estos componentes, definimos un protocolo orientado a sesión que permite al navegador y al servidor Web negociar las opciones de pago a

utilizar. Este protocolo, que hemos llamado *Extended Payment Protocol* (*EPP - Protocolo de Pagos Extendido*), también encapsula los diferentes mensajes del protocolo elegido para realizar el pago. Una vez que el pago se ha efectuado, se recibe el acceso al contenido Web por medio de una URL. Los mensajes de EPP se envían sobre un canal SSL/TLS para garantizar la confidencialidad y la integridad de la información de pago intercambiada. Este protocolo se explica en detalle en la Sección 4.2.3 y podría utilizarse para obtener el acceso, basado en pago, a cualquier recurso independientemente del protocolo de aplicación/transferencia (HTTP, FTP, etc.) usado para recuperarlo. Este protocolo intenta minimizar tanto la información a intercambiar como el número de mensajes deforma que no se introduzca una gran sobrecarga que lo haga inviable para distintos protocolos de pago, entre ellos, los de micropagos.

3. El módulo que implementa el protocolo anterior necesita utilizar distintos protocolos de pago, por lo que se define la API que cualquier wallet debería soportar para ser utilizado con EPP. Esta interfaz es independiente del protocolo de pago utilizado con el fin de facilitar el desarrollo e incorporación de componentes para nuevas propuestas de pago que se quieran incorporar a este framework. Hay que tener en cuenta, como también se mencionaba a lo largo de Capítulo 2, que en el futuro, conforme aparezcan nuevos requisitos, escenarios o nuevas primitivas criptográficas, aparecerán nuevos protocolos de pago. Por lo tanto, la API tiene que ser lo más genérica y extensible posible. Este wallet tiene que soportar tres tipos de funciones: funciones de sesión (abrir, cerrar, etc.), funciones de consulta (opciones de pago soportadas, etc.) y funciones de pago. En la Sección 4.2.4 se proporcionan más detalles acerca de las características de esta interfaz así como de sus funciones.
4. Finalmente, definimos cómo diferenciar entre aquellos enlaces que son de pago y aquellos que no lo son. Para indicar que un enlace es de pago y que, por tanto, se utilizará nuestro framework con EPP, proponemos, por cada protocolo conocido, añadir la letra “p” (de pago) antes de su nombre en el campo de protocolo de una URL, de forma similar a como se lleva a cabo para los protocolos que utilizan SSL/TLS. De esta forma tendríamos protocolos como PHTTP (pagos en HTTP), PFTP (pagos en FTP), etc. Es decir, si especificamos que un enlace tiene como protocolo PHTTP, estamos indicando que la URL apunta a un recurso que se accede finalmente por medio de HTTP y que requiere un pago que será enviado por medio del protocolo EPP (para el que se establece un canal SSL/TLS). Adicionalmente, y de forma opcional, junto con el enlace de pago podemos incluir las opciones de pago soportadas, así como los diferentes precios en las diferentes divisas. Si esta información no se incluye con el enlace, entonces sería intercambiada utilizando EPP. La forma de incluir esta información en HTML sería similar a como propone el W3C en [193], es decir, asociando los enlaces con sentencias en XML o RDF [19] que se encuentran en la sección *HEAD* de la página HTML. Los detalles de cómo se define un enlace de pago se pueden encontrar en la Sección 4.2.5.

Con los componentes que acabamos de mencionar y con las características que éstos ofrecen cubrimos las deficiencias que los frameworks existentes presentaban y que fueron analizadas en la Sección 2.3.3. En concreto, ofrecemos un framework de pagos B2C genérico que cubre todos los requisitos establecidos en la Sección 2.3.1. Como principales elementos diferenciadores frente a anteriores propuestas destacar:

- Soporta la negociación de las opciones de pago.
- Soporta distintos modelos de negocio.
- Ofrece sesiones de pago.
- Permite el intercambio de la información de pago de forma segura.
- Define la información de pago de forma genérica y puede ser utilizada tanto para describir los enlaces de pago como para intercambiarla a la hora de negociar las opciones de pago.
- Ofrece una API genérica basada en una máquina de estados para modelar cualquier protocolo.

4.2.2. Arquitectura del Navegador y del Servidor Web

En esta sección describimos los distintos componentes que hemos definido tanto para el navegador como para el servidor Web, con el fin de soportar nuestro framework de pagos por clic.

Dicho framework propone extender la arquitectura del navegador con los componentes que podemos observar en la Figura 4.2 y que comentaremos a continuación.

En primer lugar, uno o más *wallets* que serán usados para realizar la compra con distintos protocolos de pagos. Un wallet es un módulo que encapsula el comportamiento de un protocolo de pago particular. Estos wallets están definidos de acuerdo a una API genérica que es independiente del protocolo y que será comentada posteriormente en la Sección 4.2.4.

En segundo lugar, un *módulo de configuración de wallets* (*wallet config*) que permita incluir o quitar los distintos wallets que podrán ser utilizados en el navegador para realizar el pago con EPP. Así, permitimos que el usuario pueda configurar los módulos de pago con los que desee trabajar.

En tercer lugar, asociado a este último módulo, existirá un *módulo de preferencias* (*preferences*) en el que el usuario podrá especificar cuáles son sus opciones de pago preferidas (indicando su orden de preferencia), las cantidades máximas y mínimas que estará dispuesto a pagar con cada mecanismo de pago, si la elección final del protocolo es decisión del módulo EPP o si interroga al usuario, así como si desea que se le solicite o no autorización a la hora de realizar el pago.

En cuarto lugar, el *módulo EPP* que implementa el protocolo EPP. Este módulo hará uso del módulo de configuración de wallets y del módulo de preferencias de usuario para determinar qué protocolos se utilizarán en el pago. En concreto, el módulo de configuración de wallets le permitirá saber al módulo EPP los distintos wallets que hay

disponibles en un momento dado. Esta información será utilizada por el módulo EPP con el fin de determinar, en función de las preferencias configuradas mediante el módulo de configuración de preferencias, los wallets que éste soporta y, en base a los mecanismos soportados por el servidor, qué wallet será utilizado para realizar el pago. Así se facilita la elección del protocolo de pago al usuario. El usuario podría participar en esta elección (una vez que se han determinado los módulos comunes se le podría plantear al usuario que elija uno de ellos) o, por el contrario, podría estar completamente automatizada. Este comportamiento, como ya hemos explicado, se configura en el módulo de preferencias.

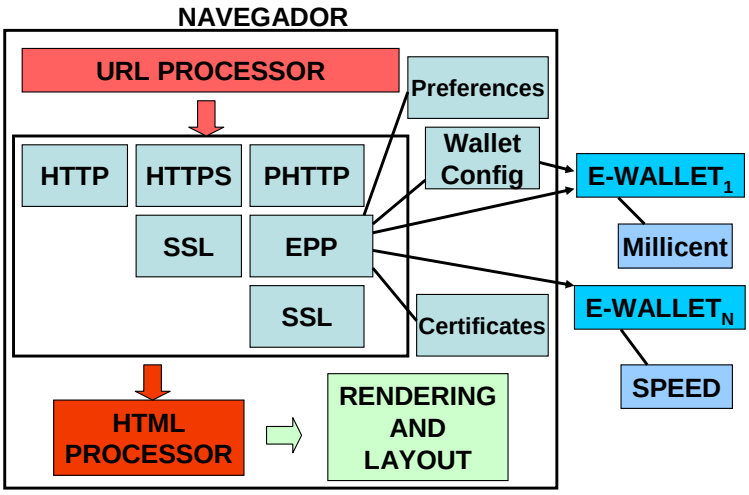


Figura 4.2. Arquitectura del navegador.

En quinto lugar, existe un *módulo de certificados* (*Certificates*) en el que el usuario puede establecer cuáles son los certificados de las CAs en las que confía para aceptar certificados de servidores de pago, así como los distintos certificados de servidores en los que confiará. Esta información será utilizada por el módulo EPP para determinar si establece o no una conexión con el servidor a la hora de realizar el pago. Este es un enfoque que permite generar confianza en el usuario y que ya ha sido puesto en práctica con los navegadores para determinar los servidores seguros a los que puede o no acceder un usuario. El módulo de certificados de pago sería similar a éste, pero restringiéndolo sólo a servidores de pago, no a cualquier servidor seguro.

Finalmente, existe un *manejador de protocolo* para cada aplicación basada en pago de acuerdo a este framework y que invoque al módulo EPP para llevar a cabo la compra (en la Figura 4.2, por ejemplo, PHTTP). En concreto, un manejador de protocolo es responsable de procesar una URL que requiera un pago para un protocolo de aplicación en particular. Por ejemplo, para aquellos enlaces que están basados y que requieran un pago usando EPP, podemos crear un manejador de protocolo llamado PHTTP como se muestra en la Figura 4.2. Cuando se hace clic en una URL que especifica este protocolo, el módulo PHTTP se invoca para efectuar el pago y obtener la URL que permitirá acceder al contenido pagado. Para este fin, el manejador PHTTP, en primer lugar, invoca al módulo EPP para realizar el proceso de pago. Una vez efectuado el pago y recibida la URL, el

manejador PHTTP invocará al módulo del protocolo HTTP para, finalmente, acceder al contenido comprado.

Los otros módulos que forman parte de la arquitectura básica del navegador son el *procesador de URLs*, (*URL processor*) que recibe de la interfaz de usuario la URL solicitada, se encarga de procesarla y determinar el manejador de protocolo que tiene que utilizar para acceder al recurso; el *procesador de HTML* (*HTML processor*), que se encarga de procesar el contenido de la página Web y el *módulo de visualización* (*rendering and layout*) que se encarga de mostrar adecuadamente el contenido recibido. Sobre estos módulos que acabamos de mencionar no ha sido necesario realizar ninguna modificación o extensión.

Desde el punto de vista del servidor, la arquitectura es la que se muestra en la Figura 4.3). En ella existen diversos módulos para los protocolos que, habitualmente, están soportados por un servidor Web, tales como HTTP y SSL/TLS. También podemos encontrar varios módulos que soportan los distintos lenguajes de programación que comúnmente se utilizan en el desarrollo de contenidos Web, como PHP, JSP y Perl. Además, para soportar nuestro framework hemos incluido el módulo EPP que implementa el protocolo EPP. Este concepto de *módulo EPP* es similar al manejador de protocolo que mencionamos anteriormente para el navegador Web. La razón por la que decidimos cambiar el nombre es para preservar la terminología que habitualmente se utiliza para nombrar los distintos componentes que forman parte de un servidor Web. Finalmente, como último componente, existe un conjunto de *módulos wallet del servidor* (*s-wallet*). Este concepto de wallet del servidor es equivalente al de wallet que presentábamos anteriormente en el navegador Web.

En la Figura 4.3, el núcleo del servidor Web (*Web Server Core*) es el encargado de recibir todas las peticiones de contenido y reenviarlas al módulo adecuado para, posteriormente, reenviar el contenido al cliente.

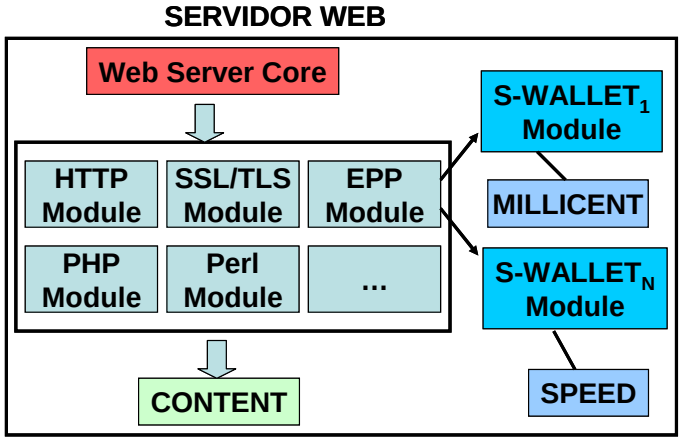


Figura 4.3. Arquitectura del servidor Web.

4.2.3. Protocolo de Pago Extendido (EPP)

En nuestro framework de pagos por clic definimos un protocolo para intercambiar información sobre las distintas opciones de pago soportadas por el cliente y el vendedor con la posibilidad de negociar el precio de estas opciones, así como para el envío y la recepción de los mensajes de pago del protocolo elegido por el cliente de entre las opciones comunes a ambos. A este protocolo hemos decidido llamarlo *Extended Payment Protocol* (*EPP - Protocolo de Pago Extendido*) y está orientado a sesión. A continuación, describimos las diferentes fases que tienen lugar en el protocolo así como los mensajes intercambiados en cada una de ellas. Posteriormente, en esta misma sección, explicaremos los diferentes modelos de pago que soporta y algunos aspectos relacionados con la seguridad.

Fases

El modo de funcionamiento de EPP está dividido en un conjunto lógico de pasos o fases. En primer lugar, hay una fase de *inicialización* donde las partes establecen un canal SSL/TLS y una nueva sesión EPP.

Antes del establecimiento de la sesión EPP, el navegador tendrá que comprobar que el servidor al que quiere acceder se trata de un servidor de pago seguro, para lo cual utilizará el módulo de certificados de pago que tiene instalado. En caso de que el certificado del servidor no se encuentre entre los certificados de servidores de pagos seguros o entre las autoridades de certificación que emiten certificados de servidores de pagos seguro, la sesión se abortará (aunque también se podría dejar a elección del usuario la acción a llevar a cabo; de esta forma, procederíamos de modo similar a como sucede actualmente con los navegadores cuando accedemos a un servidor Web del que no tenemos instalados ni el certificado de servidor ni el certificado de su autoridad de certificación).

Una vez establecido el canal SSL/TLS, para establecer la sesión EPP, básicamente, el navegador solicita el enlace de pago a acceder e intercambia con el servidor las opciones de pago soportadas. Con este intercambio básico se establece la sesión EPP. Adicionalmente, como parte de esta fase de inicialización se podrían seguir negociando las distintas opciones de pago y sus precios.

Una vez que se ha establecido la sesión y se han negociado las opciones de pago comienza la fase de *pago*. En esta fase el navegador y el servidor intercambian los mensajes de pago de acuerdo con el protocolo elegido por el módulo EPP (o por el usuario, dependiendo de sus preferencias) a partir de la respuesta del vendedor. De esta forma, EPP es responsable de encapsular los mensajes generados por el wallet del protocolo de pago correspondiente.

Una vez que finaliza el protocolo de pago, se recibe la URL que permitirá acceder al contenido que el usuario compró. Entonces, pasamos a la siguiente fase del protocolo denominada *Datos de Aplicación*, en la que se obtiene el contenido solicitado por medio del protocolo de aplicación especificado en la URL recibida en la fase de pago. Después de que el cliente obtenga el contenido, dependiendo del modelo de pago, podría visitar más enlaces de pago sin tener que efectuar pagos adicionales, como en los modelos de pago por

tiempo o pago por volumen de datos, etc (por lo que se seguiría en esta fase), o se le podría solicitar un nuevo pago. En este último caso, se retornaría de nuevo a la fase de pago.

Finalmente, después de acceder al contenido deseado, si el cliente no está interesado en acceder a contenidos de pago adicionales, puede terminar la sesión (fase de *fin de sesión*). Las fases que acabamos de mencionar aparecen reflejadas en la Figura 4.4.

Cada una de las fases que hemos explicado está compuesta, al menos, de un mensaje. Pero en algunas de ellas, se podrían intercambiar varios mensajes entre el navegador y el servidor. En general, cada intercambio entre el navegador y el servidor está basado en el patrón *petición-respuesta*, que, en general, es el patrón de intercambio básico que sigue cada protocolo, y que resulta adecuado para realizar micropagos. Además, todos los mensajes de un protocolo de pagos concreto se encapsulan usando un mensaje de EPP que puede contener un mensaje de cualquier protocolo de pagos.

Mensaje EPP

Un mensaje de EPP (*EPP Message*) está compuesto de los siguientes campos:

```
EPP message = Code, SessionID, Identifier, Data
```

donde:

- *Code* (*Código*) identifica el tipo de mensaje EPP. Para *Code* hemos definido cuatro posibles valores:
  - *Request (Petición).* Indica que es un mensaje de tipo petición.
  - *Response (Respuesta).* Indica que se trata de una respuesta a un mensaje de petición.
  - *Success (Éxito).* Este mensaje se utiliza para indicar que la sesión y el pago se llevaron a cabo correctamente.
  - *Failure (Fallo).* Este mensaje indica que se produjo un error durante el establecimiento de la sesión o en cualquier momento posterior a ésta.
- *SessionID* (*Identificador de sesión*) representa un identificador de la sesión establecida entre el navegador y el servidor. El campo *Identifier* (*Identificador*) ayuda a ligar una petición con su respuesta en una determinada sesión. Este valor se genera de forma aleatoria por el cliente en el inicio de la sesión.
- *Data* (*Datos*) se utiliza para contener la información concreta de las peticiones y respuestas usadas en las diferentes fases del protocolo (ver Figura 4.4).

A continuación, se explican en detalle las diferentes fases que hemos descrito previamente. En cada fase especificamos el contenido de las peticiones y las respuestas enviadas en el campo *Data*. El mensaje completo ya no se volverá a especificar, ya que los

campos *Code*, *SessionID* e *Identifier* son comunes a todos los mensajes, independientemente del valor del campo *Data*.

Inicialización

Tanto el navegador como el servidor intercambian cierta información que se utiliza para iniciar la sesión. El navegador inicia el protocolo usando un mensaje de petición EPP llamado *Init* (*Inicio*) (ver Figura 4.4). Su campo *Data* contiene:

Data(Init) = `Version, SupportedPaymentInfo, URL`

donde:

- *Version* (*Versión*) indica la versión del protocolo EPP que está siendo usada. Este campo se ha definido teniendo en cuenta que en el futuro pueden aparecer nuevas versiones de este protocolo.
- *SupportedPaymentInfo* (*Información de Pago Soportada*) indica las opciones de pago soportadas por el navegador. Básicamente, contiene una enumeración de los protocolos de pago, las marcas financieras y los proveedores de servicios de pago que el cliente soporta. También se pueden indicar los precios que el cliente está dispuesto a pagar con cada una de las opciones de pago. Además, el cliente puede enviar información de fidelización o credenciales que posea para obtener mejores precios. Más detalles acerca del contenido de este campo se encuentran en el Apéndice F. En concreto, este campo es un elemento del tipo *PaymentInformation*. Finalmente, el campo URL contiene el enlace de pago al que el cliente quiere acceder.

Como respuesta, el servidor envía un mensaje EPP llamado *InitResponse* (*RespuestaInicio*) cuyo campo *Data* contiene:

Data(InitResponse) = `Version, [PaymentInfo]`

donde:

- *Version* ya fue comentado en el anterior mensaje.
- Como notación, utilizamos *[Info]* para indicar que el campo *Info* es opcional. Así, el campo *PaymentInfo* (*Información de Pago*) es opcional.
- *PaymentInfo* contiene el subconjunto de opciones de pago contenidas en el campo *SupportedPaymentInfo* que el servidor soporta (este elemento también es del tipo *PaymentInformation*, para más detalles ver Apéndice F).

En este mensaje, por cada opción de pago, el servidor indica el precio del producto en las diferentes divisas soportadas. También permite indicar si el vendedor está dispuesto a negociar estos precios. En el caso de que el vendedor esté dispuesto a negociar los precios, tanto el cliente como el vendedor pueden intercambiar este mensaje tantas veces como

consideren necesario. Cuando el vendedor, en un momento dado, indica que las opciones de pago no son negociables, el cliente tiene dos opciones: o bien inicia el proceso con alguna de las opciones propuestas o finaliza la sesión. En el caso de que el cliente, en el anterior mensaje, indicara que tampoco está dispuesto a negociar, el vendedor tendría que responder con el subconjunto de precios que acepta o con un mensaje de *Failure*.

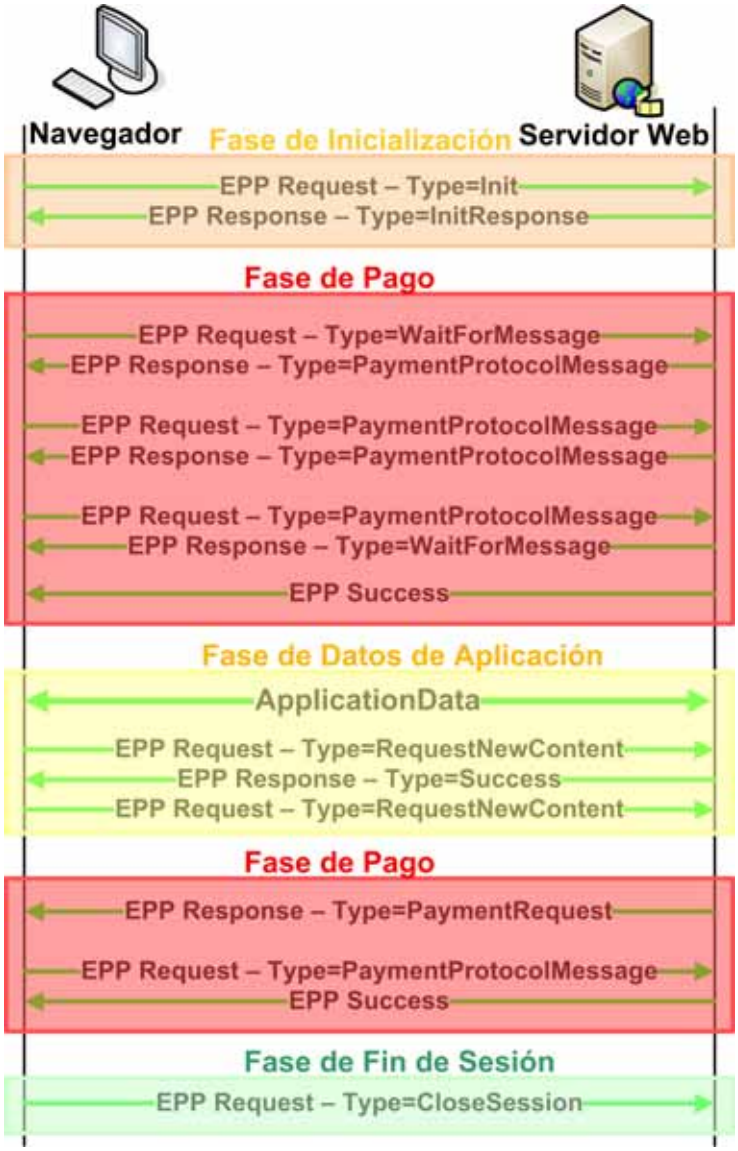


Figura 4.4. Flujo de mensajes en EPP.

Pago

En esta fase, el navegador y el servidor intercambian los mensajes del protocolo de pago que el cliente elige para la sesión. Este protocolo es elegido entre los protocolos propuestos en la fase de inicialización y que son comunes tanto a cliente como a vendedor.

Dependiendo del protocolo de pago, su primer mensaje es enviado o por el navegador o por el servidor.

Con el fin de soportar el intercambio de los mensajes del protocolo de pago, así como para encapsular dichos mensajes, hemos definido una serie de mensajes que hemos denominado: *WaitForMessage* (*Esperar Mensaje*), *PaymentProtocolMessage* (*Mensaje de Pago*) y *PaymentRequest* (*Petición de Pago*). Estos mensajes pueden ser intercambiados de diversas formas (ver Figura 4.4) que dependerán del flujo de mensajes del protocolo elegido. El número de mensajes de pago a intercambiar no está limitado en el protocolo EPP. Este límite sólo viene impuesto por el protocolo de pago elegido. El modo en que estos mensajes nos permiten modelar un protocolo de pago aparece reflejado en la Sección 4.2.4.

El mensaje *WaitForMessage* indica al receptor que el emisor está esperando un mensaje de pago del receptor de este mensaje. Definimos este mensaje porque los intercambios entre navegador y servidor están basados en el patrón petición-respuesta. Así, puede ser que en un protocolo en concreto, en un momento dado, una de las partes en dicho protocolo no le corresponda enviar un mensaje. En ese caso, se enviará este mensaje. Por ejemplo, si suponemos que tenemos un protocolo de pago que es iniciado por el servidor, el navegador envía este mensaje al servidor para solicitar el primer mensaje de pago de ese protocolo. Este mensaje EPP contiene:

```
Data(WaitForMessage) = [typeOfMessage]
```

donde:

- *typeOfMessage* (*Tipo de Mensaje*). Elemento opcional que permite indicar al emisor de este mensaje la clase de mensaje del protocolo de pagos que está esperando.

El mensaje más importante de esta fase es el mensaje denominado *PaymentProtocolMessage* (*Mensaje del Protocolo de Pagos*) que encapsula los mensajes del protocolo de pago elegido. El contenido de este mensaje se muestra a continuación:

```
Data(PaymentProtocolMessage) = Protocol, Message
```

donde:

- *Message* (*Mensaje*) contiene un mensaje del protocolo de pago elegido.
- *Protocol* (*Protocolo*) indica cuál es el protocolo de pagos del cuál viene el mensaje.

Una vez que se ha efectuado el pago, se establece la sesión y el servidor envía un mensaje EPP con el tipo *Success*. En este mensaje, el servidor proporciona una URL que el cliente utilizará para acceder al contenido solicitado.

Data(Success) = accessURL, [PaymentInfo]

donde:

- *accessURL* (*URL de acceso*) contiene una URL junto con la posible información de acceso al contenido (cookies, token de acceso, etc.) por el que se acaba de realizar el pago.
- *PaymentInfo* (*Información de pago*). Este campo opcional puede ser utilizado por el servidor para proporcionar información adicional sobre el pago que se acaba de realizar, como por ejemplo, el envío de un recibo de la transacción, proporcionar cupones de descuento o nueva información de fidelización al cliente, etc. Este campo es del tipo *PaymentInformation*.

A partir de la información proporcionada en este mensaje, el navegador puede realizar el acceso al contenido. Este acceso se lleva a cabo como se explica en la siguiente fase.

Finalmente, el último mensaje de esta fase, se utiliza para, en cualquier momento posterior al acceso a cualquier contenido, solicitar un nuevo pago y por tanto regresar a esta fase. En este caso se envía el mensaje *PaymentRequest* (*Petición de Pago*). Éste indica que el servidor requiere un nuevo pago para acceder al nuevo contenido. El contenido del campo *Data* es:

Data(PaymentRequest) = PaymentInfo

donde:

- *PaymentInfo* ya fue previamente explicado (ver el mensaje *InitResponse*).

Con este mensaje, tanto el navegador como el servidor regresan a esta fase de pago para intercambiar los mensajes del protocolo de pago elegido. Estos mensajes se intercambian por medio de los mensajes que acabamos de explicar en este apartado. Así el cliente consigue el acceso a un nuevo contenido.

Datos de Aplicación

El cliente accede al contenido utilizando el protocolo de aplicación establecido en la URL obtenida (HTTP, FTP, etc.). Una vez que el cliente ha accedido al contenido, la sesión puede finalizar o, dependiendo del modelo, el cliente podría querer acceder a otros enlaces de pago (ver Sección 4.2.5).

Cuando la sesión está establecida, el mensaje EPP que se utiliza para solicitar el acceso al nuevo contenido se llama *RequestNewContent* (*Petición de Nuevo Contenido*). Su campo de datos contiene:

Data(RequestNewContent) = [SupportedPaymentInfo], URL

donde:

- *SupportedPaymentInfo* ya fue explicado anteriormente en el mensaje *Init*.
- *URL* indica el enlace de pago al que el usuario quiere acceder.

Si el pago que se llevó a cabo en la anterior fase da derecho a acceder a más de un enlace (como en los modelos pago por un conjunto de enlaces, pago por tiempo, etc), en este caso, el servidor envía una respuesta de *Success* que contiene la URL del recurso a acceder, tal y como se comentó anteriormente.

Por el contrario, si el pago previo sólo da acceso a un único contenido, el servidor envía un mensaje de *PaymentRequest* y se regresa de nuevo a la fase de pago.

Fin de la sesión

Una vez que el cliente ha accedido al contenido, éste puede cerrar la sesión, si es que no está interesado en más contenidos con el mismo servidor. Para este propósito hemos definido un mensaje EPP denominado *CloseSession* (*Cerrar Sesión*). De esta manera, navegador y servidor puede liberar los recursos asociados a la sesión.

Modelos de pago

Como se ha mencionado a lo largo de la sección anterior, nuestro framework es capaz de soportar distintos modelos de negocio. El modelo, y su información asociada, pueden ser especificados tanto en la información de pago que se puede adjuntar con el enlace (en la Sección 4.2.5 se explicará cómo), como en la fase de negociación de EPP (en el campo *PaymentInfo*). Los distintos modelos que se pueden especificar son: pago por cada enlace o el pago por un conjunto de enlaces. La especificación está abierta a cualquier modelo nuevo que pueda aparecer como podría ser el pago por tiempo o por datos.

En general, los vendedores requerirán un pago por cada enlace. Por tanto, el usuario inicia la sesión, efectúa el pago y, finalmente, obtiene el contenido. Después de este proceso, si el cliente solicita un nuevo enlace de pago en la misma sesión, se le pedirá que realice un nuevo pago (tal y como se muestra en la Figura 4.4, en la segunda fase de pago por medio del mensaje *PaymentRequest*). El proceso de pago por este nuevo enlace es más rápido porque la fase de inicialización ya fue ejecutada. En caso contrario, se iniciaría una nueva sesión.

Otra posibilidad es que tanto el cliente como el vendedor acuerden el pago por un conjunto de enlaces. En este modelo, el cliente efectúa un pago inicial que le permite acceder a un conjunto de enlaces (el número de enlaces al que el usuario puede acceder es especificado en el campo de *PaymentInformation* o por medio de la información de pago asociada al enlace). El pago se efectúa cuando el usuario accede al primer elemento del conjunto de enlaces. De esta manera, hay una fase de inicialización, a continuación, una fase de pago y, después, una fase de datos de aplicación. Para los siguientes enlaces, no se ejecuta la fase de pago y al usuario se le proporciona directamente el nuevo enlace solicitado mediante la fase de datos aplicación. En este modelo, el control de los enlaces de pago visitados lo lleva a cabo el servidor. Si el cliente quisiera acceder a más enlaces de

pago, el servidor enviará un mensaje *PaymentRequest* indicando que se requiere un nuevo pago.

Consideraciones de seguridad

Desde el punto de vista de seguridad, es importante destacar que la seguridad de la fase de pago depende del protocolo de pago elegido ya que nuestra propuesta solamente encapsula los mensajes del protocolo elegido.

En EPP hemos definido que el intercambio de mensajes entre el cliente y el servidor esté basado en el uso de un canal SSL/TLS con el propósito de evitar que algún tercero pueda ver o cambiar la información de pago intercambiada entre el navegador y el servidor. Por ejemplo, un tercero podría estar interesado en modificar la información sobre los protocolos de pago soportados para que tanto el navegador como el servidor eligieran un protocolo de pagos débil que pudiera ser atacado. Con SSL/TLS se garantiza la confidencialidad y la autenticación del servidor, por tanto el cliente está seguro de que está pagando al servidor deseado. SSL/TLS también ofrece mecanismos para proteger la información de la sesión. Por tanto, esta protección la podríamos extender a EPP.

Además, con respecto a los servidores de pago que serán de confianza, así como las Cas que se consideran seguras para la emisión de certificados de servidor de pagos, se sigue un modelo similar al que actualmente soportan los navegadores Web, de forma que el cliente puede especificar en su navegador cuáles son los servidores de pago que son de confianza. De esta forma, el navegador podrá rechazar de forma automática (en el caso de que el cliente lo configure así, también se podría interrogar al usuario para que sea éste el que tome la decisión final) el realizar la compra a servidores de dudosa reputación o en los que el cliente tuvo anteriormente problemas.

Finalmente, tanto si el usuario como el vendedor desean garantizar la autenticidad, integridad y no repudio de la información intercambiada, pueden enviar la información de pago firmada por medio de una firma XML. Para este fin, en los elementos definidos (*SupportedPaymentInfo* y *PaymentInfo*, que son del mismo tipo) se ha incluido un elemento opcional que puede contener dicha firma.

4.2.4. Wallet genérico

Una vez que hemos diseñado un framework que es independiente del protocolo de pago específico a utilizar, necesitamos definir un mecanismo para incorporar el soporte de un protocolo específico en esta propuesta de una forma genérica.

Como veíamos en la sección donde presentábamos la arquitectura de navegadores y servidores Web (Sección 4.2.2), a los componentes que soportan el pago con un determinado protocolo los denominamos wallet o wallet de servidor, respectivamente. A partir de ahora, cuando utilicemos el término wallet será para referirnos de manera indistinta a cualquiera de los dos. Es importante definir este componente de forma genérica con la finalidad de que, en cualquier momento, pueda ser incorporado un nuevo sistema de pagos al navegador o al servidor, sin tener que llevar a cabo modificaciones en el software desarrollado. De esta forma, tendríamos una interfaz común para los

desarrolladores de software de pago que oculte las particularidades de cada método de pago y que facilite la extensibilidad del sistema y la confianza del usuario en el framework (como mencionan Gefen et al. [118]), ya que la forma de incluir cualquier sistema de pago será siempre igual.

Este modelo está siendo utilizado con éxito para la incorporación de otros componentes, como son el soporte de criptografía en dispositivos hardware en los navegadores. En este sentido, están las propuestas de PKCS#11 [247] para navegadores basados en Mozilla y de CryptoAPI [195] para Internet Explorer.

Nuestro objetivo es intentar modelar los componentes de pago de forma similar, independientemente de las características particulares. Para este propósito, en primer lugar, hemos realizado una revisión en la literatura de los principales protocolos de pago (Millicent [117], Payword [238], SET [274], Paycash [218], etc.), muchos de los cuales podemos encontrarlos descritos en el Capítulo 2; así como las propuestas que presentamos en el Capítulo 3 (SPEED [249] y PURSE-COIN [251, 252]). A continuación, se han analizado las propuestas de APIs surgidas hasta ahora, como era la de SEMPER [166, 220] o la de IOTP [32, 82] (ver Sección 2.3.2). Estas APIs, desde nuestro punto de vista, no son lo suficientemente genéricas, utilizan bastante información del protocolo de la capa superior y no cubren algunos aspectos como son la gestión de sesiones. Nuestro objetivo es definir un wallet de tal forma que su API pueda ser utilizado tanto en nuestra propuesta como en cualquier otra propuesta genérica de pagos.

A continuación, en los siguientes apartados mostraremos cómo el wallet se ha diseñado a partir de una máquina de estado que intenta representar a cualquier protocolo, presentaremos las funciones que soporta el wallet y, finalmente, presentaremos un ejemplo del uso del wallet mostrando la interacción que tendría lugar entre el wallet y el cliente.

Máquina de estados del wallet

Conceptualmente, en cualquier protocolo de pagos se produce un intercambio de mensajes entre dos o más partes. Habitualmente, las principales entidades son el cliente y el vendedor. Adicionalmente podría participar un PSP, un banco, un broker, etc. Aunque alguna de estas entidades participe, al final, el pago habitualmente se realiza entre el cliente y el vendedor. De esta forma, el diseño del wallet se realizará teniendo en cuenta que éste es el principal intercambio y, que si alguna de las otras partes participa, será el wallet el que encapsule los intercambios con estas terceras entidades.

Si pensamos en el intercambio que se produce entre el cliente y el vendedor, el wallet es un mecanismo de abstracción y lo podríamos pensar como una caja negra que recibe mensajes que procesa y, como respuesta a esos mensajes, creará uno o varios mensajes (ver Figura 4.5). Es importante puntualizar aquí que el wallet no es el responsable de recibir el mensaje a través de la red. Esta función la realizará otro componente, por ejemplo, el servidor Web. Una vez que el servidor Web ha recibido el mensaje se lo pasará al wallet. Así cuando mencionamos que el wallet recibe un mensaje, estamos diciendo que se invoca a una función de su API que se encargará de procesar un mensaje de un protocolo.

Partiendo de estas premisas hemos diseñado la máquina de estados de Figura 4.5. Cuando una de las partes inicia la ejecución del protocolo, según su rol, puede ocurrir tres acciones: que sea la responsable de crear el primer mensaje, que tenga que esperar la recepción de un mensaje para procesar o que tenga que esperar a un determinado evento (una acción de algún tercero) antes de crear o procesar un mensaje.

En un momento dado, cuando un wallet procesa/crea un mensaje o esperar un evento, la siguiente acción a realizar se puede modelar con 4 estados: crear un mensaje de respuesta, esperar a recibir otro mensaje para procesarlo, esperar un evento, finalización del protocolo de forma correcta o error en la ejecución del protocolo. Con estos estados y estas transiciones podemos modelar los protocolos de pago y derivar las funciones que serían necesarias para este wallet. Las funciones las presentaremos en el siguiente apartado.

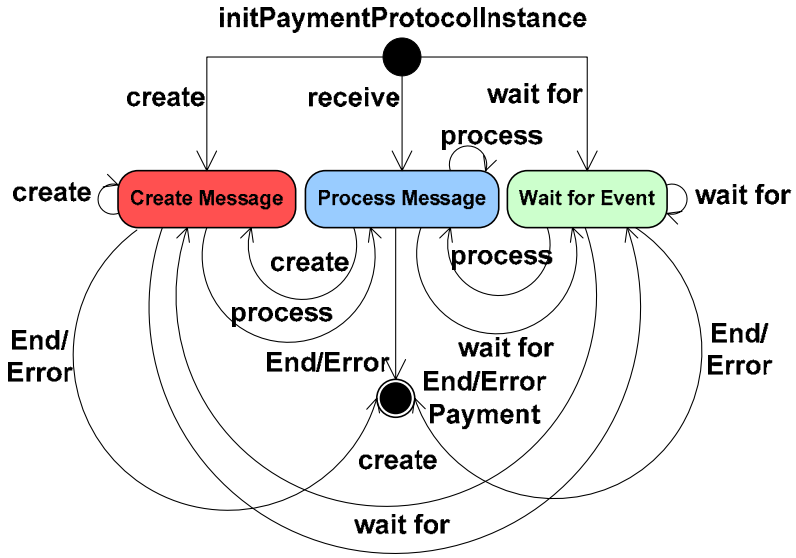


Figura 4.5. Máquina de estados del wallet.

Funciones del wallet

A partir de esta máquina de estados que hemos definido y que representa a cualquier protocolo hemos definido las funciones del wallet. Estas funciones las podemos dividir en tres grandes grupos: funciones de pago, funciones de gestión de la sesión y funciones de consulta.

Funciones de pago

Las principales funciones que deberá tener todo wallet para poder llevar a cabo el pago son: *inicialización*, *procesar mensaje*, *crear mensaje*, *esperar evento*, *error pago* y *fin pago* (ver Tabla 4.1). Cada una de estas funciones como resultado devolverá el siguiente estado al que tiene que pasar el wallet y a qué función hay que invocar.

Cada uno de los estados y funciones está representado en la Figura 4.5, donde cada estado del diagrama representa a una función y cada una de las transiciones representa el resultado de invocar a esa función. Este grupo de funciones constituye lo que denominamos el conjunto de funciones de pago. Es importante destacar que estas funciones también nos permiten modelar otros protocolos relacionados con pagos, como pueden ser devoluciones, intercambios de valor, carga de monederos, etc.

Nombre de la función	Descripción de la función
initPaymentProtocolInstance	Función que inicializa un determinado protocolo de pago.
createMessage	Función que crea el siguiente mensaje del protocolo de pagos.
processMessage	Función que procesa un mensaje que hemos recibido de la otra parte.
waitForEvent	Función que espera a que se produzca un determinado evento en la ejecución del protocolo.

Tabla 4.1. Funciones de pago del wallet.

Un wallet, además de las funciones relacionadas directamente con el proceso de pago, necesita otros tipos de funciones como funciones de gestión de sesión y funciones de consulta. Estas las comentaremos en los siguientes apartados.

Nombre de la función	Descripción de la función
openSession	Función que inicializa una sesión con el wallet.
stopSession	Función que para el actual intercambio de mensajes de la sesión.
resumenSession	Función que reanuada una sesión que ha sido interrumpida.
closeSession	Función para finalizar una sesión y liberar los recursos asociados.

Tabla 4.2. Funciones de sesión del wallet.

Funciones de gestión

Las funciones de gestión de la sesión se utilizan para establecer una sesión con el wallet, de forma que se permita realizar uno o más pagos utilizando los mecanismos de pago soportados por tal wallet. Para este fin se han definido una función de inicio de sesión, que permite la inicialización del wallet, así como una función de cierre de sesión que permite liberar los recursos asociados al wallet. Cuando se inicia la sesión, se obtiene un

identificador asociado a la sesión ya que es posible que podamos ejecutar en paralelo distintas sesiones. Este identificador de sesión se utilizará en la invocación del resto de funciones a la hora de realizar el pago (ver Tabla 4.2).

Nombre de la función	Descripción de la función
getProtocols	Permite obtener los protocolos soportados por el wallet. Por ejemplo, SPEED, PURSE-COIN, etc.
getBrands	Retorna las marcas financieras soportadas (Visa, Mastercard, etc.).
getPaymentServiceProviders	Permite obtener los proveedores de servicios de pago soportados (Broker SPEED, Banco X, etc.).
getPaymentOptions	Retorna las opciones de pago soportadas (combinaciones de los tres elementos anteriores).
getCurrencies	Retorna las divisas soportadas (Euros, Dólares, etc.).
getBalance	Permite obtener información acerca del saldo disponible con una determinada opción de pago.

Tabla 4.3. Funciones de consulta del wallet.

Funciones de consulta

También hemos definidos varias funciones de consulta para obtener información sobre los distintos parámetros relacionados con los métodos de pago soportados por el wallet. Por tanto, hemos definido una función para obtener las distintas marcas financieras soportadas, otra para obtener los distintos protocolos soportados, otra para consultar las distintas divisas que soportan las entidades y protocolos anteriores, así como una función para conocer el saldo actual del que dispone el usuario para efectuar pagos.

Interacción del wallet en el framework

A continuación, en la Figura 4.6, mostramos la interacción que tiene lugar entre el wallet y los distintos elementos del sistema a la hora de realizar el pago. Por simplificar la explicación de esta interacción se explicará desde el punto de vista del cliente y no se proporcionará el funcionamiento al completo del servidor. El comportamiento del servidor sería similar al mostrado para el cliente. Para esta interacción supondremos que el cliente quiere obtener un recurso al que finalmente se accede por medio de HTTP. Los pasos son:

1. El cliente hace clic sobre la URL deseada en el navegador que provoca que se invoque al módulo PHTTP (mensaje 1 en la Figura 4.6. En esta descripción todos los mensajes estarán referidos a esta figura). Este módulo recibe como parámetros la URL de pago y, opcionalmente, la información de pago que se adjunta con el enlace (ver Sección 3.5). En este caso, por simplificar el diagrama, hemos supuesto que, en el cliente, el módulo PHTTP y EPP están integrados.

2. El módulo EPP comprueba los distintos wallets que están instalados y les pregunta acerca de los distintos métodos de pago que soportan (con el método *GetPaymentMethods* – mensajes 2 y 3. Las funciones del wallet mencionadas en estos pasos se encuentran descritas en el anterior apartado). Tal y como se ha diseñado el wallet, un mismo módulo podría soportar la implementación de distintos protocolos de pago. Esto se ha hecho con el fin de agrupar protocolos que ofrecen características similares o que incluso pueden compartir un mismo elemento para realizar el pago como podría ser una cuenta bancaria o el uso de una tarjeta inteligente de unas características específicas. Para este ejemplo supondremos que cada wallet soporta únicamente un protocolo de pago. Así, este método retorna las distintas opciones de pago soportadas. Supongamos que el $Wallet_1$ soporta Millicent, siendo la marca financiera el propio vendedor, y el $Wallet_N$ soporta el pago con SPEED utilizando como marca financiera *Monedero WG10*.
3. Por cada opción de pago, se comprueba el saldo disponible (método *GetBalance*, mensajes 4 y 5). En el caso de que se reciba información del enlace de los métodos soportados por el vendedor, sólo se interrogará a aquellos que son comunes. En este ejemplo, hemos supuesto que sólo hay disponible una opción de pago por wallet.
4. El módulo EPP construye el mensaje *Init* (ver Sección 4.2.3) conteniendo la información obtenida de los distintos wallets y lo envía al servidor (mensaje 6).
5. Una vez que el vendedor recibe el mensaje y lo procesa, comprueba los distintas opciones de pago que soportan sus wallets de servidor y, en el mensaje *InitResponse* (mensaje 7), envía las que son comunes a ambos. Supongamos también que en este mensaje el vendedor ha indicado que no permite la negociación de las opciones de pago y precios ofrecidos.
6. Una vez que el módulo EPP del cliente recibe el mensaje *InitResponse*, elige la opción de pago que más le interesa y establece una sesión con el wallet elegido (*OpenSession* en el mensaje 8), en este ejemplo supondremos que ha sido el $Wallet_1$.
7. Una vez establecida la sesión, invoca al método que permite iniciar el proceso de pago (*InitPaymentProtocolInstance* en el mensaje 9) que indicará si el cliente tiene que crear, recibir o esperar un mensaje conforme a la máquina de estados del protocolo, tal y como se mostró anteriormente. En el ejemplo, como resultado de invocar este método, el wallet indica que tiene que pasar a crear un mensaje (como decíamos anteriormente utilizamos Millicent, un protocolo de micropagos que necesita dos mensajes para llevar a cabo el pago y dónde el proceso lo inicia el cliente, véase Capítulo 2).
8. El módulo EPP invoca al wallet para la creación del mensaje de pago (*CreateMessage*, mensaje 10). Este método, a su vez, indica que la acción a realizar será procesar un mensaje que tendrá que recibir del servidor un mensaje para procesar.
9. El módulo EPP con el mensaje obtenido del wallet, lo encapsula dentro del mensaje *PaymentProtocolMessage* (mensaje 11) y lo envía al servidor.

10. El servidor procesa el mensaje *PaymentProtocolMessage*, invocará al wallet correspondiente y como resultado obtendrá un mensaje del wallet que se encapsula dentro del mensaje *PaymentProtocolMessage* (mensaje 12) y se envía al cliente.

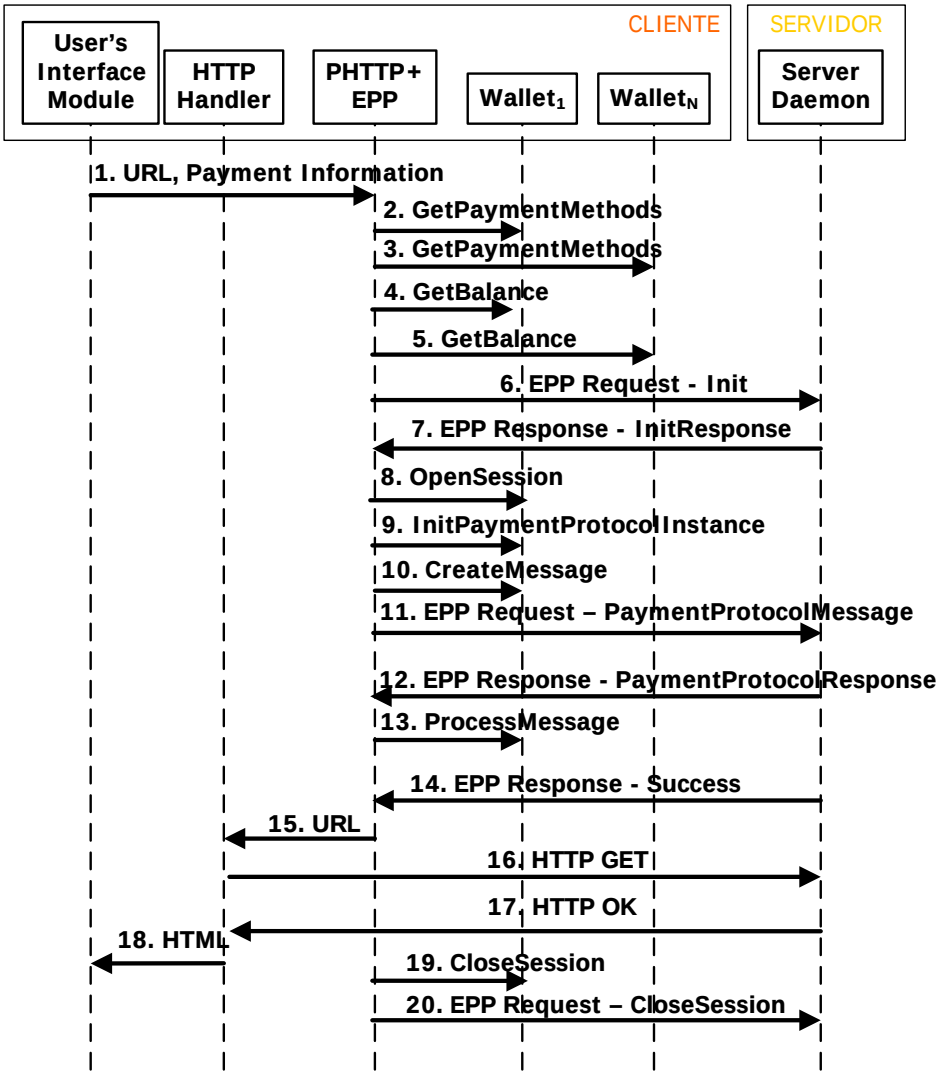


Figura 4.6. Interacción de todos los elementos de EPP para un pago.

11. Cuando el módulo EPP recibe el mensaje de pago, lo envía al wallet (*processMessage*, mensaje 13) y, una vez procesado, el wallet indica que el proceso de pago ha terminado.

12. El módulo EPP espera a la recepción del mensaje de *SUCCESS* que el wallet genera justo después de crear el mensaje de pago que acaba de recibir el cliente. En este mensaje (mensaje 14), el módulo EPP recibe la URL que permitirá acceder al contenido por el que el cliente pagó.
13. En este ejemplo, la URL para acceder al contenido como es una URL que será accedida con el protocolo HTTP, el módulo EPP invoca al módulo HTTP (mensaje 15).
14. El módulo HTTP finalmente recupera el contenido por medio del método *GET* de HTTP (mensaje 16), recibiendo en el método *OK* (mensaje 17) la correspondiente página HTML a visualizar.
15. La página recibida se muestra al usuario (mensaje 18).
16. Finalmente, si no se van a realizar más pago, la sesión con el wallet y con EPP finaliza (mensajes 19 y 20).

4.2.5. Enlaces de pago

En esta sección, describimos cómo definir los enlaces de pago en una página HTML. Cuando se hace clic sobre uno de estos enlaces, se lanza el proceso de pago descrito anteriormente.

En HTML, proponemos referenciar a este protocolo por medio de URLs. Este protocolo será referenciado añadiendo una "p" a cada protocolo conocido en el que estamos interesados soportar pagos. Por ejemplo, para referenciar a una página Web que se accede por medio de un pago, la referenciaremos usando la cadena *phttp://PáginaWeb* (es decir, ejecuta EPP para realizar un pago por un recurso que se recupera por medio de HTTP).

Cada nuevo protocolo está asociado con un manejador de protocolo en el navegador Web. Este manejador es responsable de ejecutar el protocolo EPP, efectuando el pago con el wallet adecuado. Después de que el pago se haya efectuado, el módulo pasa la URL al navegador para acceder al contenido utilizando el manejador de protocolo correspondiente (al módulo HTTP para el ejemplo comentado).

Opcionalmente, en una URL podemos incluir información sobre los protocolos de pago, las marcas financieras y los precios. Si esta información no está incluida con la URL, el cliente no puede saber el/los protocolo(s) soportado(s) para llevar a cabo el pago hasta que la fase de inicialización de EPP se haya ejecutado.

En la página Web, la información de pago se incluye en la sección *HEAD* de HTML por medio de XML o sentencias RDF. La URL referencia a esta información a través de un atributo *id* en la etiqueta *A*, que es la etiqueta que se utiliza en HTML para especificar enlaces. El proceso seguido para definir un enlace basado en pago es el siguiente:

1. Incluimos un enlace en la página Web que representará al enlace de pago. Para este propósito utilizamos la etiqueta *A*. Entonces, en el atributo *href* de la etiqueta *A* incluimos el URL del contenido que está basado en pago, por ejemplo, *<A href="phttp://recurso">*. Es importante señalar que aunque la URL del enlace de pago sea *phttp://recurso*, no quiere decir que el enlace que finalmente obtiene el

usuario y que le permite acceder al contenido será *http://recurso*. El mecanismo utilizado para generar la nueva URL así como el tiempo durante el cual el usuario puede acceder a la URL, dependerá de la implementación del servidor.

2. Identificamos de manera unívoca la URL por medio del atributo *id* de la etiqueta *A*.
3. En la sección *HEAD* de la página HTML incluimos la información de pago que referencia a este enlace por medio del atributo *id*. Esta información está expresada por medio de sentencias XML o RDF que indican los protocolos de pago soportados, las marcas soportadas, los PSPs soportados así como el precio de cada combinación de estos elementos.

A continuación mostramos un ejemplo de una página Web que contiene un enlace de pago para esta tesis (Figura 4.7). En concreto, el enlace es *phttp://server/directory/thesisARM.pdf*. Básicamente, este enlace solicita un pago por un fichero PDF que contiene la tesis. El vendedor de esta tesis se llama *Research Papers Ltd* y este enlace cuesta *15 Euros* utilizando el protocolo de pagos *SPEED*. Este pago sólo da derecho a acceder a este documento, por esta razón, en la información de pago incluida se indica como modelo de pago seguido *one payment one use* (*un pago un uso*). En este caso, es el propio vendedor el que actúa como PSP, por lo tanto, el proceso de pago se llevará a cabo directamente con el vendedor.

Cuando se realiza un clic sobre el enlace de pago, el manejador del protocolo PHTTP se lanza y, entonces, el protocolo EPP se ejecuta. Una vez que el protocolo es ejecutado, el módulo EPP obtiene una URL que permite acceder al producto. Entonces, el módulo PHTTP invocará al módulo HTTP para se muestre en el navegador y que permita descargar el PDF elegido..

La definición de los distintos elementos que se necesitan para expresar la información de pago fue realizada por medio de XML, como vemos en la Figura 4.7. Para este fin, hemos creado un esquema XML que proporciona los elementos necesarios para describir las marcas financieras, los protocolos de pago, el modelo a seguir, los precios, etc. La definición de estos elementos, que se describe en el Apéndice F, se ha realizado de la forma más genérica posible de manera que se pueda reutilizar en otros escenarios como los que presentaremos en el siguiente capítulo.

Aparte de este esquema, hemos creado otro que define los distintos elementos que forman parte del protocolo EPP y que, a su vez, referencia al que contiene los elementos básicos. A partir de estos esquemas hemos generado el código fuente necesario para procesar los distintos elementos, tal y como se explica en la siguiente sección. Adicionalmente, para mejorar la descripción de la información de pago, estos elementos podrían ser anotados con información semántica [159]. Este tipo de información facilita el descubrimiento de información y su procesamiento automático basado en agentes inteligentes o herramientas de procesamiento automático [69, 128, 181].

```
<HTML>
<HEAD>
<TITLE>Example per-fee-link</TITLE>
   <PaymentInformation   xmlns="urn:umu:paymentframework:paymentinformation:v0.1"
                              xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
                              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ID="idpaperepp">
        <PaymentProtocols>
                <PaymentProtocol ID="ID1" ProtocolID="urn:umu:speed:v0.1">
                        <Name><InternationalString lang="en-gb">SPEED</InternationalString></Name>
                        <Version>0.1</Version>
                </PaymentProtocol>
        </PaymentProtocols>
        <Brands>
                <Brand ID="ID2" BrandID="WG10">
                        <Name><InternationalString lang="en-gb">Monedero Euro6000</InternationalString></Name>
                        <Logo>http://www.euro6000.com/wg10.jpg</Logo>
                </Brand>
        </Brands>
        <PaymentServiceProviders>
                <PaymentServiceProvider ID="ID3" PSPID="http://www.researchpapers.com">
                        <Name><InternationalString lang="en-gb">Research Papers Ltd</InternationalString></Name>
                        <Logo>http://www.researchpapers.com/logo.jpg</Logo>
                        <PaymentURL>https://www.researchpapers.com/paymentURLSPEED</PaymentURL>
                        <VendorID>http://www.researchpapers.com</VendorID>
                </PaymentServiceProvider>
        </PaymentServiceProviders>
        <Prices>
                <Price ID="ID4">
                        <Amount>15</Amount>
                        <Currency ID="ID5">
                        <CurrencyCodeType>
                        urn:umu:paymentframework:paymentinformation:v0.1:currency:ISO4217-A
                        </CurrencyCodeType>
                        <Code>EUR</Code>
                        </Currency>
                </Price>
        </Prices>
        <PaymentDescriptions>
                <PaymentDescription ID="ID6" negotiable="false">
                        <PaymentReference>phttp://server/directory/idreference/thesisARM.pdf</PaymentReference>
                        <PaymentProtocolsRefs>
                                <PaymentProtocolRef>#ID1</PaymentProtocolRef>
                        </PaymentProtocolsRefs>
                        <BrandsRefs>
                                <BrandRef>#ID2</BrandRef>
                        </BrandsRefs>
                        <PaymentServiceProvidersRefs>
                                <PaymentServiceProviderRef>#ID3</PaymentServiceProviderRef>
                        </PaymentServiceProvidersRefs>
                        <PricesRefs>
                                <PriceRef>#ID4</PriceRef>
                        </PricesRefs>
                        <PaymentModel ID="ID7">
                                <PaymentModelCodeType>
                                urn:umu:paymentframework:paymentinformation:v0.1:paymentmodel:onepaymentoneuse
                                </PaymentModelCodeType>
                        </PaymentModel>
                        <Expiration>2008-12-31T09:30:47-05:00</Expiration>
                </PaymentDescription>
        </PaymentDescriptions>
        <Expiration>2008-12-31T09:30:47-05:00</Expiration>
   </PaymentInformation>
</HEAD>
<BODY>
        <A href="phttp://server/directory/thesisARM.pdf" id="#idpaperepp">Thesis ARM</A>
</BODY>
</HTML>
```

Figura 4.7. Ejemplo de página HTML con enlace de pago.

4.2.6. Implementación

En esta sección describimos la implementación que hemos realizado de este framework de pagos. Es importante destacar que nuestra propuesta no requiere re-implementar todos los navegadores y servidores Web para soportarla. Por el contrario, nuestra propuesta

puede ser incluida en la mayoría los navegadores y servidores Web conocidos ya que éstos ofrecen mecanismos de extensibilidad genéricos para soportar nuevos protocolos. Así, conseguimos que el framework goce de una mayor aceptabilidad que si fuéramos al desarrollo de un nuevo navegador que incorpora estas características o si implicara que tuviéramos que introducir cambios en los ya existentes.

El soporte de un nuevo protocolo en Firefox se lleva a cabo por medio de lo que es conocido como *extensión de manejador de protocolo* (*protocol handler extension*) [198], mientras que en Internet Explorer el soporte de un nuevo protocolo se realiza por medio de lo que se conoce como *Protocolos Asíncronos Conectables* (*asynchronous pluggable protocols*) [194]. Para nuestra implementación la opción elegida fue el desarrollo de un manejador de protocolo para Firefox.

Desde el punto de vista de los servidores, uno de los servidores Web más conocidos, el servidor Apache, puede ser extendido por medio de *módulos*. De hecho, en este servidor, el soporte de SSL/TLS está implementado por medio de un módulo. De forma similar, nuestro protocolo se implementa como un módulo de filtrado a nivel de conexión. Más detalles acerca de la arquitectura del servidor Apache se pueden encontrar en [154].

Para estos componentes ha sido necesario, como elemento fundamental, el desarrollo del protocolo EPP. Además, para realizar las pruebas de implementación del protocolo ha sido necesario la utilización de diversos protocolos de pago. Para este propósito, hemos hecho uso de la implementación de Millicent [117] y SPEED (ver Capítulo 3). Millicent es un protocolo de micropagos que es adecuado para realizar el pago de contenidos de muy bajo valor, como un Euro o menos. SPEED es un protocolo más seguro y que garantiza el no repudio y el intercambio equitativo. El producto se entrega cifrado y está diseñado para realizar el pago con el monedero electrónico de las tarjetas inteligentes. Los detalles de SPEED se pueden encontrar en el Capítulo 3.

Las implementaciones de los protocolos de pago fueron desarrolladas en C++ bajo el sistema operativo Linux. Los mensajes de ambos protocolos están expresados en ASN.1 ya que, antes del éxito de XML, todos los protocolos estaban definidos por medio de ASN.1. Incluso a día de hoy, ASN.1 en determinados casos puede ser más interesante que XML tal y como veremos posteriormente. Los detalles acerca de la implementación de SPEED se pueden encontrar en el Capítulo 3. La implementación de Millicent se desarrolló para estas pruebas de implementación.

Para la implementación de los protocolos, como librería de funciones criptográficas y de definición y manejo de las estructuras ASN.1 utilizamos OpenSSL [211], en concreto, la versión 0.9.8g de ésta. Esta librería es de las más completas en cuanto a la funcionalidad ofrecida, se distribuye bajo licencia GPL y está ampliamente comprobada.

Para el protocolo EPP hemos creado también una librería en C++. A partir de esta librería hemos desarrollado un módulo cliente que será utilizado con el manejador que hemos creado para PHTTP en Firefox. Para el servidor, hemos desarrollado un módulo de servidor para Apache.

Para la implementación de EPP hemos desarrollado código a partir de OpenSSL para establecer el canal SSL/TLS. En esta implementación de EPP todos los mensajes han sido codificados en XML utilizando XML Schemas y, para la generación del código a partir de

los esquemas, se ha utilizado el generador de código LMX XML Schema C++ [58]. La razón por la que nos decidimos por esta utilidad es porque es simple de utilizar, es rápida y fácil a la hora de manipular las estructuras XML y el código generado es portable. El hecho de codificar los mensajes en XML es porque estamos interesados en que este protocolo también se pueda utilizar para el pago de servicios Web y sus mensajes puedan ser fácilmente enviados sobre SOAP [119]. De forma adicional, también hemos desarrollado los mensajes de la fase de pago en ASN.1 con OpenSSL versión 0.9.8g para realizar la evaluación del protocolo y comparar con la implementación de XML tal y como se describe en la siguiente sección.

4.2.7. Evaluación

El propósito de esta sección es analizar algunos aspectos relacionados con la viabilidad de la implementación del framework propuesto. Nuestro objetivo es evaluar si nuestra propuesta puede ser utilizada con diferentes protocolos de pago por medio de los distintos componentes que hemos presentado en el framework.

A partir de la implementación desarrollada, cuyos detalles fueron descritos en la anterior sección, hemos realizado una serie de pruebas en el sistema. En las pruebas, lo que queremos es medir el incremento que supone introducir el protocolo EPP para el transporte de los mensajes de pago y comprobar si es posible asumir tal incremento para conseguir un framework que facilite los pagos en la Web a las diferentes partes (clientes, vendedores, desarrolladores de sistemas de pagos, etc.). También compararemos las diferencias que existen en realizar la implementación de EPP con ASN.1 y con XML.

Aunque en general la implementación de un protocolo en ASN.1 es más eficiente que en XML [199] queremos comprobar si la implementación XML es factible y así tener una implementación del protocolo que pueda ser utilizada con servicios Web. Las pruebas se han realizado en una red local con tres equipos con las características que se muestran en la Tabla 4.4.

Características	Valor
Procesador	Intel Core 2 Duo
CPU MHz	2.66 Ghz
Memoria RAM	2 Gb
Sistema Operativo	Fedora Core 10
Kernel	Linux 2.6.27.12-170.2.5
Compilador	gcc 4.3.2

Tabla 4.4. Características de las máquinas de pruebas para EPP.

A continuación, comentaremos los resultados obtenidos en las pruebas. Cada una de las pruebas realizadas fue repetida 100 veces y los resultados presentados en las siguientes figuras están basados en el cálculo de un intervalo de confianza del 95% para la mediana

(ver Tabla 4.5). El intervalo de confianza se calcula a partir del valor la mediana y el valor que aparece a lado de cada tiempo. Por ejemplo, el intervalo de confianza al 95% para el protocolo EPP XML con Millicent se encuentra entre 41,34-0,15 y 41,34+0,15 ms.

En primer lugar presentaremos los resultados obtenidos con Millicent, posteriormente presentamos los resultados obtenidos con SPEED.

Protocolo	Tiempo (ms)	IC 95%
Millicent	1,10	±0,00
EPP ASN.1 (fase de pago) con Millicent	1,29	±0,01
EPP XML (fase de pago) con Millicent	41,34	±0,15

Tabla 4.5. Millicent vs EPP con Millicent.

En la Figura 4.8, comparamos el tiempo de ejecución del protocolo de pago Millicent con el tiempo de ejecución de encapsular los mensajes de Millicent en la fase de pago de EPP implementada en ASN.1. Es decir, enviar los *EPP Request* y *Response* conteniendo en el campo *PaymentProtocolMessage* los mensajes de Millicent.

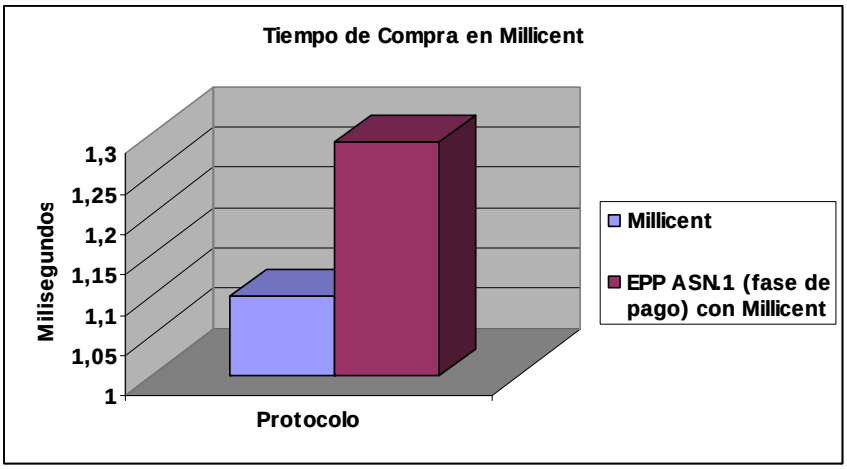


Figura 4.8. Tiempo de ejecución de Millicent vs Millicent con EPP en ASN.1.

El objetivo es determinar la sobrecarga introducida por la implementación ASN.1 de EPP por lo cual sólo comparamos esta fase. No comparemos el tiempo de ejecución de Millicent con el tiempo de ejecución total de EPP porque Millicent no soporta la negociación de las opciones de pago ni la entrega del producto. Los tiempos utilizados y los intervalos de confianza al 95% aparecen en la Tabla 4.5. En esta situación la sobrecarga introducida por EPP es del 17'5%. Es decir, encapsular los mensajes de Millicent en EPP sólo supone un 17,5%.

En la Figura 4.9, comparamos la diferencia entre la implementación de la fase de pago de EPP tanto en ASN.1 como en XML. Como se puede observar, la implementación de ASN.1 es 32,1 veces mejor que la implementación de XML. Sin embargo, sólo representa

un incremento de 40,05 milisegundos (es decir, 0,04 segundos). Por tanto, podríamos decir que la demora introducida es aceptable desde el punto de vista del usuario.

A continuación pasamos a realizar el mismo tipo de análisis con el protocolo SPEED. Los resultados que se presentarán en las siguientes figuras parten de los valores e intervalos de confianza que se muestran en la Tabla 4.6. En esta tabla para cada producto aparece la mediana del tiempo en milisegundos en llevar a cabo la transacción. A continuación, a aparece el valor que permite determinar el intervalo de confianza al 95% para la mediana.

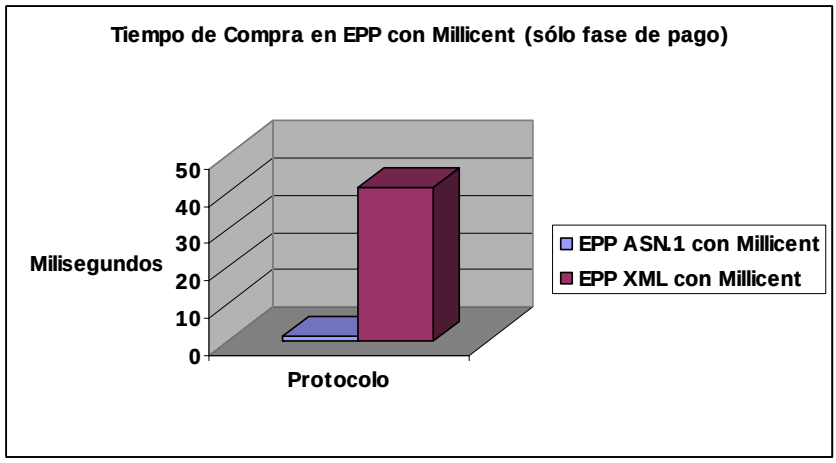


Figura 4.9. Tiempo de EPP ASN.1 vs EPP XML con Millicent.

Gráficamente estos resultados son los que aparecen en la Figura 4.10. Del análisis de esta comparación se desprende que la sobrecarga introducida por la versión ASN.1 de EPP con SPEED oscila entre un 10,7% (0,07 seg.) y un 38,96% (0,1 seg.). En cambio, la versión XML de EPP varía entre un 1,84% (0,04 seg.) y un 304,95% (2,06 seg.). Como se puede apreciar en la Figura 4.10, la implementación de EPP en XML, conforme aumenta el tamaño la diferencia con la implementación de ASN.1 se hace mayor.

Existen varias razones que explican estas diferencias [199]:

- El tamaño de los datos del elemento ASN.1 puede ser tres veces más pequeño que el de XML.
- En el tiempo de transmisión, el déficit de tiempo de XML puede elevarse hasta 13.5 veces por un elemento y el tiempo de codificar/decodificar datos puede llegar hasta diez veces.
- En ASN.1 si es posible enviar información binaria, en cambio, en XML requiere que ésta sea codificada en Base64, con lo cuál el tamaño de información a enviar es mayor.

De esta forma, en la implementación de EPP con XML cuando mayor sea el tamaño del producto, mayor será la sobrecarga. Este hecho puede producir problemas de escalabilidad

cuando hay muchas solicitudes concurrentes. A pesar de este hecho, las empresas están usando XML y los servicios Web de forma intensiva [94, 147], a diferentes niveles, para diferentes tipos de servicios y propósitos, como se menciona en [213], porque "proporcionan una infraestructura de computación distribuida para la colaboración e integración de aplicaciones tanto intra- como entre-empresas".

	32 bytes	IC 95%	12 Kb	IC 95%	281 Kb	IC 95%	3.7 Mb	IC 95%
SPEED	204,98	±1,83	207,85	±2,78	224,08	±3,74	675,35	±3,25
EPP ASN.1 SPEED	284,85	±0,37	285,54	±0,91	324,84	±0,95	747,94	±4,87
EPP XML SPEED	208,75	±6,04	217,31	±3,76	395,46	±3,14	2734,88	±4,50

Tabla 4.6. SPEED vs EPP SPEED.

Para mitigar los problemas de escalabilidad de XML, se ha propuesto varias soluciones como el uso de técnicas de redundancia, balanceo de carga y soluciones basadas en cluster [27, 212]. De esta forma, podríamos conseguir una mayor disponibilidad y rendimiento como se propone en [27, 199]. Otras opciones adicionales para hacer frente a este problema de rendimiento son servicios Web basados en ASN.1 (Fast Web Services – Servicios Web Rápidos) [266], XML comprimido [288], o parsers XML basados en streaming [322]. A pesar de estas soluciones, si el rendimiento fuera crítico frente a la interoperabilidad, entonces podríamos utilizar la implementación ASN.1 que supone una menor sobrecarga cuando hay que enviar contenidos electrónicos de cierto tamaño, como se ha señalado anteriormente (ver Figura 4.10).

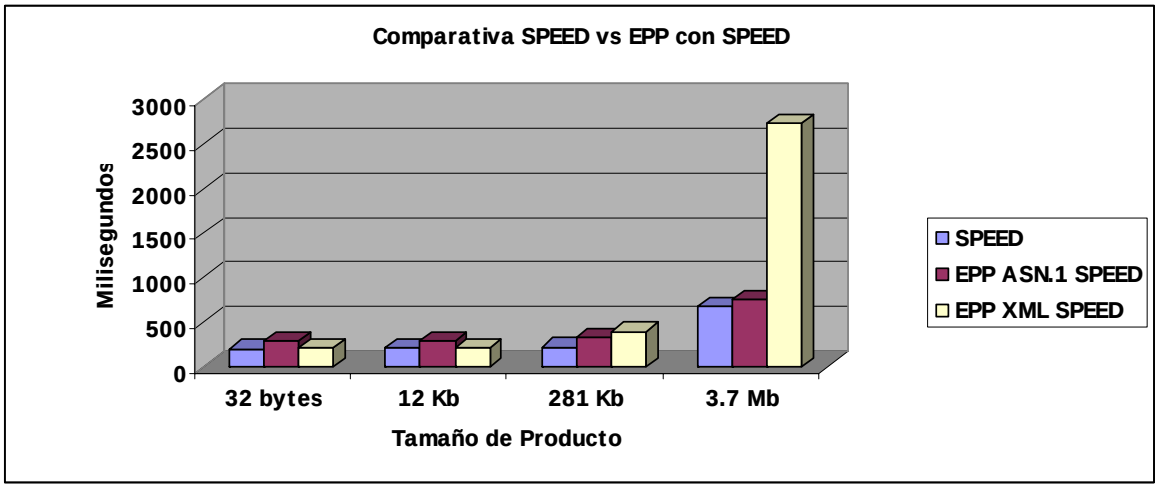


Figura 4.10. SPEED vs EPP con SPEED.

Hasta ahora sólo hemos mostrado el tiempo de ejecución de la fase de pago. A continuación, presentamos la Tabla 4.7 donde mostramos el tiempo de ejecución de EPP con SPEED (en milisegundos), indicando el tiempo consumido en cada una de las fases para distintos tipos de productos (32 bytes, 12 Kbytes, 281 Kbytes, y, 3.7 Mbytes). Estos tiempos aparecen representados gráficamente en la Figura 4.11. En ese gráfico, como se puede observar, el tiempo de la fase de fin de sesión es insignificante. También es importante mencionar que con SPEED, la fase de *Datos de Aplicación* no es necesaria porque el producto se entrega con el pago.

	EPP Inic.	IC 95%	EPP Pago	IC 95%	EPP Fin Sesión	IC 95%
32 bytes	16,01	±0,07	284,85	±0,37	3,19	±0,01
12 Kb	16,07	±0,02	285,54	±0,91	3,04	±0,03
281 Kb	16,11	±0,03	324,84	±0,95	3,31	±0,01
3.7 Mb	15,94	±0,02	747,94	±4,87	5,87	±0,01

Tabla 4.7. Tiempos de EPP con SPEED por fases.

De estos resultados podemos derivar varias conclusiones. En primer lugar, las fases de inicialización y de fin de sesión de EPP introducen una sobrecarga casi constante en milisegundos debido a que la información intercambiada, es decir, los protocolos de pago negociados, es la misma para todos los productos. Sin embargo, la sobrecarga que estas fases suponen a la sobrecarga total de EPP es proporcionalmente menor al aumentar el tamaño del producto ya que el tiempo de la fase de pago se incrementa, como podemos ver en el último caso de la Figura 4.11. Esta sobrecarga no la podemos evitar ya que necesitamos establecer el protocolo de pago a utilizar.

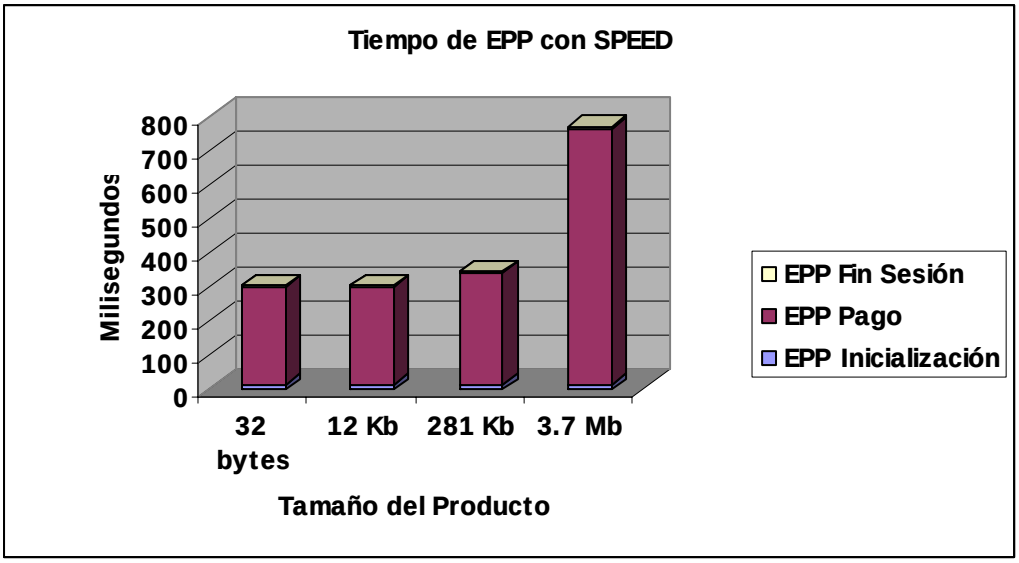


Figura 4.11. Tiempo de las fases de EPP con SPEED.

En segundo lugar, la sobrecarga introducida por EPP en sus dos variantes (ASN.1 y XML) es de apenas unos milisegundos (entre 4 y 100 milisegundos) salvo para el caso de un producto de 3.7 MB en la implantación de EPP con XML (en este caso 2 segundos). Anteriormente, hemos mencionado las razones que explican este incremento y se han propuesto varias soluciones para hacerle frente. Por tanto, en general, podemos decir que la sobrecarga introducida por EPP se podría considerar aceptable de cara a ofrecer una solución que facilite la elección y el uso de distintos protocolos de pago para realizar compras en la Web.

Finalmente, podemos afirmar que es posible desarrollar este framework para soportar el uso de diferentes protocolos de pago para efectuar el pago de contenidos en la Web. De hecho, no hubo problemas que impidieran el desarrollo del protocolo EPP así como los protocolos definidos en este framework como wallets. De esta forma hemos podido comprobar que efectivamente el diseño de wallets basado en una máquina de estados finitos nos ha permitido implementar diferentes protocolos, en este caso Millicent y SPEED, para incorporarlos al framework de forma transparente a éste.

4.2.8. Conclusiones

El modelo de pagos por clic es un modelo interesante para llevar a cabo la adquisición de contenidos en la Web. Sin embargo, como analizamos, hasta ahora, aparte de que las soluciones definidas no eran lo suficientemente completas, no se daban las condiciones necesarias para su uso, como por ejemplo, protocolos suficientemente seguros que permitan generar confianza en los usuarios, suficiente extensión del comercio electrónico, etc. Conforme el uso del comercio electrónico se extiende y, a su vez, la necesidad del uso de distintos protocolos de pagos, surgen la necesidad de ofrecer un framework de pagos para la adquisición de contenidos Web. En respuesta a esta necesidad, hemos propuesto un framework de pagos por clic.

Nuestra propuesta está basada en la definición de varios componentes. En primer lugar presentamos una serie de componentes a añadir en las arquitecturas de los navegadores y servidores Web. Los componentes se han definido teniendo en cuenta los mecanismos de extensibilidad de los navegadores y servidores Web. Por tanto, no requiere la re-implementación de éstos y, por lo tanto, facilita el proceso de aceptación e incorporación.

A continuación, hemos definido un protocolo de pagos genérico orientado a sesión, que permite la negociación y elección del protocolo de pago a utilizar (y los precios asociados) y que soporta distintos modelos de negocio, como pagos por un solo uso, pago por un conjunto de enlaces, etc. Además, este protocolo podría ser utilizado para efectuar el pago de cualquier contenido que pueda ser referenciado por medio de una URL. De esta forma se facilita que clientes y vendedores puedan utilizar distintas soluciones de pago.

Asociado a este protocolo genérico y para facilitar el uso de distintos protocolos de pago con éste, hemos definido una API genérica para wallets. La definición de esta API está basada en una máquina de estados finitos que permite modelar cualquier protocolo de pagos. Incluso se podría crear un wallet para aquellas transacciones que se realizan con una tarjeta de crédito por medio de SSL/TLS con una entidad financiera. Así, incluso la

solución más utilizada a día de hoy por los usuarios podría ser incorporada a este framework. Este diseño del wallet permite, además de soportar los protocolos de pago, soportar otros protocolos relacionados con éstos, tales como devoluciones, carga de monederos, etc. Con este componente, además, se facilita que la incorporación de nuevos sistemas de pago sea uniforme para clientes y vendedores, de forma que estén seguros de que están llevando a cabo el proceso correctamente.

Finalmente, hemos definido los elementos XML necesarios para describir la información de pago que será intercambiada en los mensajes del protocolo genérico así como nos permitirá definir los enlaces de pago. Al definir esta información e incorporarla a la información Web además se facilita el descubrimiento, procesamiento, procesos de comparación, etc de la información de pago a través de motores de búsqueda y agentes inteligentes.

Los enlaces de pago se incluirán en las páginas Web cuando se quiera solicitar el pago de un producto. Estos enlaces permiten describir la información relacionada con el pago como el precio, protocolos soportados, modelos de pago a seguir, etc. De esta forma, se permite especificar para distintos productos distinta información de pago. Además se uniformiza la manera en la que el usuario obtiene la información de pago y le permite distinguir claramente cuáles son los productos de pago. A partir de estos enlaces se lanzará el proceso de pago basado en los componentes anteriormente definidos y que le permitirán al usuario elegir el método de pago a utilizar y realizar el pago de forma independiente al protocolo particular que desee utilizar.

Por tanto, podemos afirmar que ofrecemos una solución global y completa que contempla, de forma genérica, todos los elementos que se necesitan para soportar el modelo de pago por clic. Con esta solución podemos proporcionar a los usuarios, vendedores y desarrolladores de sistemas de pago un sistema uniforme que les genere confianza. Además, los componentes al ser definidos de forma genérica podrían ser utilizados para otros entornos. Así, el protocolo genérico de pagos, los wallets y la forma de expresar la información de pago permitiría su adaptación a otros escenarios como el pago basado en el uso de recibos, la compra de la utilización de servicios Web o su uso con agentes inteligentes.

Aparte del diseño de los distintos componentes, también hemos presentado una implementación de la propuesta realizada. Ésta nos ha servido para realizar distintas pruebas con dos protocolos de pago de diferentes características. Estas pruebas nos permiten concluir que el framework que proponemos es viable y que la sobrecarga introducida por el protocolo podría ser aceptada por las distintas partes. Así, esta solución junto con el uso de protocolos de pago como SPEED podría facilitar que los usuarios confíen en los sistemas de pago para la Web.

4.3. Servicios Web de pago con WSDL y UDDI

4.3.1. Introducción

Los servicios Web facilitan la interoperabilidad porque son independientes de la tecnología subyacente. Esta es la principal razón por la que muchas compañías han iniciado un proceso de migración para proporcionar sus servicios como servicios Web. Este hecho también ha fomentado el desarrollo de nuevos servicios como servicios Web porque el número potencial de usuarios puede ser considerable. Por ejemplo, Google ya ofrece algunos servicios como servicios Web, tales como Google Search, Adwords, etc.

Los servicios Web están descritos por medio de WSDL. Estas descripciones normalmente están publicadas en la Web o en repositorios tales como UDDI. Además, normalmente, el acceso a estos servicios es gratuito. Sin embargo, cada vez más compañías están desarrollando servicios Web basados en pago porque éstas quieren obtener beneficios de los servicios que proveen. Por ejemplo, el servicio Google Adwords está basado en este enfoque y dependiendo de la operación a llevar a cabo el precio es diferente.

El principal problema que aparece en estos servicios es que, a día de hoy, no está definida la información de pago asociada a los servicios que permita automatizar el descubrimiento y procesamiento de los servicios, sus precios, comparar entre los distintos vendedores, etc. A día de hoy el mecanismo propuesto para expresar esta información es mediante políticas y acuerdos a nivel de servicio (SLAs – Service Level Agreements). Por medio de Web Service Level Agreements (WSLA) y WS-Agreement se podría negociar los mecanismos de pago a utilizar. Estas propuestas por el contrario no son adecuadas para publicar sólo la información de pago ni han definido cómo incluir esta información en los repositorios UDDI. Además, en estas políticas tampoco se ha definido cómo describir esta información de pago. Así como tampoco se ha definido cómo llevar a cabo el pago. Este aspecto está abierto. Tal y como se menciona en [292] la información de pago se podría enviar en la cabecera de los mensajes SOAP (Simple Object Access Protocol – Protocolo de Accesso a Objetos Sencillo). Sin embargo, esto imposibilita el uso de protocolos de pago que requieran más de un mensaje.

Por esta razón, los proveedores de servicios Web han optado por incorporar esta información de pago en páginas Web. En estas páginas la información se describe en lenguaje natural, es decir, sin utilizar algún lenguaje procesable por una máquina que facilite su procesamiento automático. Por tanto, el descubrimiento de servicios y sus requisitos es más complejo y no puede ser automatizado por motores de búsqueda o por agentes inteligentes como se propone en [102, 162]. Es más, la comparación de precios o la negociación de servicios con características con similares características es un proceso complejo. Otro de los problemas que se presenta es cómo realizar el pago de estos servicios Web facilitando el uso de distintos mecanismos de pago.

Como solución a estos problemas, en esta sección presentamos un framework de pagos para servicios Web. El objetivo de este framework es satisfacer los objetivos planteados al principio de este capítulo para el pago por el acceso a servicios Web. Es decir, que se trate de una solución genérica, que soporte distintas opciones de pago, que permita la

negociación de éstos, etc. Para intentar solucionar este problema se podría intentar expresar esta información por medio de enlaces de pago utilizando el framework planteado en la anterior sección. Sin embargo, este modelo no es adecuado ya que la información de pago que se puede expresar para un servicio no sólo depende de una posibilidad sino que podría depender de las operaciones, interfaces y bindings a utilizar. Además, la descripción de servicios Web no se suele publicar en la Web sino en repositorios como UDDI.

Este framework de pagos está basado en dos propuestas. En primer lugar nos basaremos en *Payment Annotations for Web Service Description Language* (*PA-WSDL – Anotaciones de Pago para el Lenguaje de Descripción de Servicios Web*). PA-WSDL, que será introducida en esta sección, describe cómo incorporar la información de pago junto con la descripción del servicio en WSDL para aquellos servicios Web que requieran un pago para poder acceder. Para la definición de PA-WSDL nos basaremos en los elementos definidos para expresar información de pago en la Sección 4.2. Es importante destacar, que con PA-WSDL no estamos especificando el precio de los diferentes productos que pueden ser comprados/accedidos invocando al servicio.

En segundo lugar nos basaremos en el protocolo EPP que fue presentado en la Sección 4.2 para la negociación y selección del precio de las opciones de pago así como de la ejecución del protocolo de pagos elegido. Debido a que el protocolo EPP ya fue descrito anteriormente nos centraremos en explicar PA-WSDL y del protocolo EPP sólo mencionaremos cómo utilizarlo durante el proceso de pago.

En PA-WSDL, básicamente, proponemos dos modos de incluir la información de pago en WSDL: extendiendo directamente WSDL o por medio de WS-Policy. Se define también cómo incluir la información de pago en la especificación de UDDI para facilitar el descubrimiento de servicios, así como la información de pago asociada a estos servicios que están anotados con información de pago.

A continuación, en la siguiente sección presentaremos una visión genérica del framework de pagos para servicios Web. Posteriormente, proporcionaremos una descripción de PA-WSDL en detalle.

4.3.2. Visión genérica del framework pagos para servicios Web

Nuestro objetivo es describir la información de pago asociada a un servicio Web de pago de una forma que sea procesable por una máquina y, a continuación, poder realizar el pago para poder acceder al servicio deseado. Además, queremos incluir la información de pago junto con la descripción Web del servicio para facilitar la búsqueda automática y las posibles negociaciones.

La Figura 4.12 muestra el escenario genérico de este framework. El primer paso es la publicación del servicio Web junto con la información de pago.

Por simplificar, supongamos que la clase de servicio que el usuario quiere sólo puede ser accedido si se realiza un pago. En estas circunstancias, cuando un cliente quiere hacer uso de un tipo de servicio específico, éste (o un agente inteligente que lo represente) lleva a cabo una búsqueda en la Web o en un servidor UDDI para descubrir los proveedores de servicio ofreciendo esa clase de servicio (paso 2). Como respuesta, el cliente obtiene, del

servidor UDDI, la lista de los proveedores de servicio (paso 3). A continuación, el cliente solicita la descripción WSDL del propio proveedor del servicio (paso 4). La información sobre las diferentes ligaduras y operaciones también puede ser obtenida del servidor UDDI (paso 5).

A partir de la descripción WSDL y su información de pago, el cliente puede iniciar el proceso de selección para determinar el proveedor del servicio a utilizar de acuerdo a sus preferencias personales. Por ejemplo, el cliente podría descartar aquellos proveedores de servicios que requieran el uso de protocolos de pago no soportados por el usuario (paso 6). Así, usando el conjunto completo de proveedores de servicio, el cliente puede comparar precios y otra información de pago (paso 7). A continuación, el cliente elige el proveedor del servicio que ofrece las mejores condiciones (paso 8).

El cliente comienza el proceso de pago con el proveedor de servicios (paso 9) utilizando alguna de la información asociada a WSDL por medio de la extensión que hemos definido, como precios, credenciales, etc.

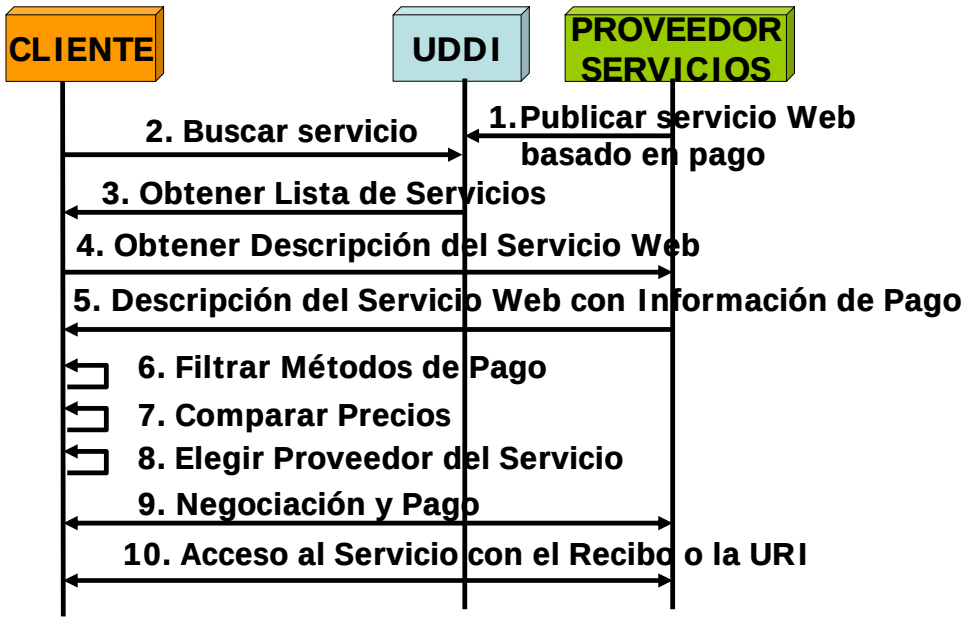


Figura 4.12. Proceso de pago en un escenario de servicios Web de pago.

El mecanismo de pago podría ser llevado a cabo de dos maneras distintas. La primera sería un enfoque similar al propuesto por Jennings et al. para el pago de servicios SIP en el que el cliente realiza el pago a un PSP (o al vendedor). Éste a cambio proporcionará un recibo al cliente que le permitirá acceder al servicio. Este recibo debería ser una sentencia basada en XML. El recibo codifica sentencias tales como “E pagó X Euros a un proveedor de servicio SP a través de un PSP P para acceder a la operación O del servicio S”. En este caso el PSP podría ser incluso el propio proveedor del servicio.

La otra alternativa para realizar el pago sería por medio del protocolo EPP (descrito en el Capítulo 4). En este caso, en la descripción WSDL del servicio, la URL de acceso al

servicio representaría una URL de pago. Es decir, para acceder al servicio, el cliente Web, por medio de EPP, efectuaría el pago y, de esta manera, obtendría la URL así como el material necesario que realmente le permitiría acceder al servicio, por ejemplo, un recibo como el que se acaba de describir (paso 9). La ventaja de este último enfoque es que además de soportar cualquier protocolo de pagos, permitiría la negociación del precio del acceso al servicio y el envío de los mensajes de pago.

Después del pago, el cliente puede invocar el servicio (paso 10). Todos los pagos realizados al proveedor del servicio a través del PSP se reciben en la cuenta del proveedor del servicio. En el caso del PSP que provee de recibos, el rol del PSP podría se llevado a cabo o por el proveedor del servicio o por un banco o por otra entidad.

A continuación, explicaremos en detalle cómo incluir la información de pago en WSDL. La descripción de cómo utilizar EPP se puede encontrar en el Capítulo 4.

4.3.3. Payment Annotations for WSDL (PA-WSDL)

A la hora de adjuntar la información de pago en WSDL podemos seguir dos enfoques. El primer enfoque es extender WSDL para incluir directamente la información de pago. El segundo enfoque es utilizar la extensión WS-Policy para WSDL para incluir la información de pago como un requerimiento. El modo en el que incluimos la información de pagos en ambos enfoques es lo que constituye *Payment Annotations for WSDL* (*PA-WSDL – Anotaciones de Pago para WSDL*).

El primer enfoque es adecuado cuando el proveedor del servicio no soporta WS-Policy o cuando no quiere añadir un nivel de complejidad adicional para expresar la información de pago.

El segundo enfoque se propone para aquellos proveedores de servicios que ya soportan WS-Policy para expresar otros requisitos relacionados con seguridad, fiabilidad, etc. De esta forma, la información de pago se expresaría como otro requisito adicional. En los siguientes apartados describimos cómo incluir la información de pago en ambos enfoques.

Extensión de WSDL

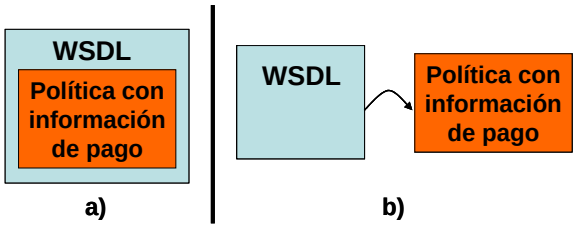
Existen dos alternativas a la hora de incluir o asociar la información de pago a WSDL. En la primera alternativa, esta información se incluye directamente junto con la descripción WSDL en el mismo fichero (ver Figura 4.13a). La otra alternativa pasa por que la información de pago se encuentre en un documento separado y se haga referencia a ésta desde el WSDL (ver Figura 4.13b). De esta forma la información de pago se puede reutilizar en distintos WSDLs.

En ambos casos, al final, la información de pago se expresa por medio del elemento *PaymentInformation* así como por uno o varios conjuntos de elementos que son referencias a descripciones de pago (que se encuentran dentro del elemento *PaymentInformation*, ver Apéndice F).

El elemento *PaymentInformation* contiene la información que describe los protocolos de pago, las marcas financieras, los PSP soportados así como los precios a pagar. También

contiene las diferentes descripciones que indican la necesidad de realizar un pago para acceder al servicio, independientemente de si el pago se requiere para una operación, una ligadura o un servicio.

En la primera alternativa este elemento se inserta al mismo nivel que los elementos tales como *interface*, *binding* y *service* (ver Figura 4.13a) de WSDL. Para la segunda alternativa hemos definido un atributo denominado *PaymentInformationURI* para referenciar al documento donde se encuentra este elemento. Una vez obtenido este elemento, éste se puede utilizar para llevar a cabo el proceso de filtrado de información determinando los protocolos, marcas financieras soportadas y precios.



a) Información de pago incluida en WSDL. b) Información de pago referenciada desde WSDL.

Figura 4.13. La información de pago en WSDL.

Para especificar la información de pago asociada a *operations*, *interfaces*, *bindings* y *services*, utilizamos el elemento *PaymentDescriptionsRefs* o el atributo *PaymentDescriptionRefsRef* (una referencia al elemento que acabamos de mencionar).

El elemento *PaymentDescriptionsRefs* se define como un conjunto de referencias a elementos *PaymentDescription* que se incluyen en el elemento *PaymentInformation* que se ha obtenido por alguno de los enfoques antes mencionados. Este elemento se encuentra o bien dentro de los elementos operations, interfaces, binding o services; o bien al mismo nivel que el elemento *PaymentInformation* si se ha utilizado el atributo *PaymentInformationURI*.

El atributo *PaymentDescriptionsRefsRef* es una referencia al elemento *PaymentDescriptionsRefs* y se utiliza cuando es posible reutilizar ese conjunto de descripciones para varios elementos dentro del mismo WSDL.

Queremos destacar que si especificamos la información a nivel de operación, el precio de la operación es el mismo independientemente del *binding* utilizado con la interfaz en la que la operación está descrita. Esto nos permite tener, para la misma interfaz, operaciones que requieren un pago para acceder y operaciones que no lo requieren.

Este mismo enfoque se utilizaría para especificar la información de pago a nivel de interfaz, de binding y servicio. Con la diferencia de que en los *bindings* podemos también tener operaciones con precios diferentes. En este caso, esta información sería incluida en el elemento *operation* que está en el elemento *binding*.

Si especificamos la información de pago a distintos niveles, la cantidad a pagar es la suma de todas las cantidades expresadas en los distintos niveles.

En la Figura 4.14 mostramos un ejemplo de cómo se incluiría la información de pago en WSDL para el esquema que aparece en la Figura 4.13a. En negrita aparecen resaltados los distintos elementos insertados.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:description …
xmlns:pInf="urn:umu:paymentframework:paymentinformation:v0.1"
targetNamespace="urn://smswebservice.um.es">
 <pInf:PaymentInformation>
        <pInf:PaymentProtocols> … </pInf:PaymentProtocols>
        <pInf:Brands> … </pInf:Brands>
        <pInf:PaymentServiceProviders> … </pInf:PaymentServiceProviders>
        <pInf:Prices> … </pInf:Prices>
        <pInf:Credentials> … </pInf:Credentials>
        <pInf:LoyaltyInformations> … </pInf:LoyaltyInformations>
        <pInf:PaymentDescriptions>
                <pInf:PaymentDescription negotiable="true" ID="IDPD1" receipt="false">
                …
                </pInf:PaymentDescription>
        </pInf:PaymentDescriptions>
        <pInf:AdditionalInformation> …. </pInf:AdditionalInformation>
        <pInf:Expiration>2009-12-17T09:30:47.0Z</pInf:Expiration>
        <ds:Signature Id="IDSig"> . . . </ds:Signature>
 </pInf:PaymentInformation>
 <pInf:PaymentDescriptionsRefs … ID="IDCPD1" >
        <pInf:PaymentDescriptionRef>#IDPD1</PaymentDescriptionRef>
        …
 </PaymentDescriptionsRefs>
 <wsdl:types> … </wsdl:types>
 <wsdl:message name="NewSMSMessageRequest"> ... </wsdl:message>
 …
 <wsdl:portType name="SMSPortType"> … </wsdl:portType>
 <wsdl:binding name="SMSBinding" type="tns:SMSPortType"
        pInf:PaymentDescriptionsRefsRef="#IDCPD1">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        ...
        </wsdl:binding>
        <wsdl:service name="SMSService">
                <wsdl:port name="SMSPort" binding="tns:SMSBinding">
                        …
                </wsdl:port>
        </wsdl:service>
</wsdl:description>
```

Figura 4.14. WSDL con información de pago conforme a PA-WSDL.

En primer lugar, con el elemento *description* aparece el espacio de nombres del esquema que define la información de pago conforme se especificó para EPP. A continuación, dentro del elemento *description* aparece un elemento *PaymentInformation*. Este elemento describe toda la información de pago asociada al acceso a los servicios que se definen en ese WSDL. Finalmente, podemos ver cómo se ha especificado el atributo *PaymentDescriptionsRefsRef* del elemento *binding*. Este atributo referencia al conjunto de descripciones de pago (*IDCPD1*) que se encuentra a continuación del elemento *PaymentInformation*. Este conjunto a su vez hace referencia a las distintas descripciones

de pago que indican el precio, las opciones de pago soportadas, etc. En este ejemplo podemos ver cómo se referencia a la descripción de pago *IDPD1*. De esta forma estamos indicando que el precio para todas las operaciones de ese *binding* es el mismo.

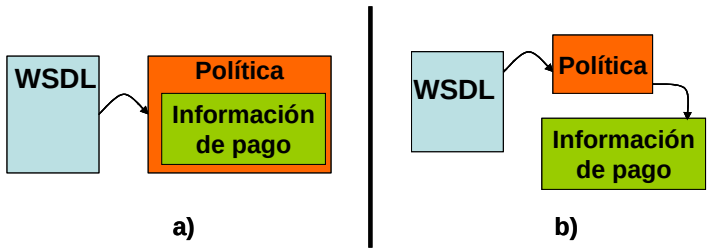
El resto de casos de cómo incorporar la información no se mostrarán debido a las similitudes en cuanto a la información que incluir. Por ejemplo, para el caso de la Figura 4.13b el WSDL tendría el mismo aspecto salvo que en vez de que el elemento *PaymentInformation* estuviera contenido dentro del WSDL estaría referenciado mediante el atributo *PaymentInformationURI*. Para los casos que se explican a continuación las diferencias también serían mínimas como se podrá apreciar en la descripción que viene a continuación.

Información de pago en WS-Policy

La información de pago se puede incluir en WS-Policy como un requerimiento adicional. De esta forma podemos combinar requerimientos de diferentes dominios (seguridad, mensajes fiables, etc) de una manera uniforme facilitando de este modo su posterior procesamiento.

WS-Policy también permite incluir las políticas en varios niveles en WSDL. Por tanto, podemos incluir esta política y así, la información de pago, a nivel de mensaje, de operación, de ligadura y de servicio.

Para la información de pago, como sentencias de políticas, incluimos el elemento *PaymentDescriptionRef* en lugar del conjunto de elementos *PaymentDescriptionRef* (es decir, al elemento *PaymentDescriptionsRefs*) ya que en WS-Policy se recomienda el uso de la forma normal donde cada sentencia está incluida en un elemento *Policy* diferente. Esta referencia apunta a un elemento *PaymentDescription*. Este elemento está incluido en el elemento *PaymentInformation*. En este caso, este último elemento puede ser incluido de diversas formas. La primera posibilidad es como una extensión WSDL al mismo nivel que el elemento *service* (ver Figura 4.13a). La segunda posibilidad es incluirlo dentro del elemento raíz del elemento *Policy*, es decir, extendiéndolo (ver Figura 4.15a). Finalmente, este elemento podría estar almacenado en otro lugar, siempre y cuando pueda ser referenciado (ver Figura 4.15b).



a) WSDL referencia a una expresión de política extendida con información de pago. b) La expresión de política referencia a información de pago en otro documento.

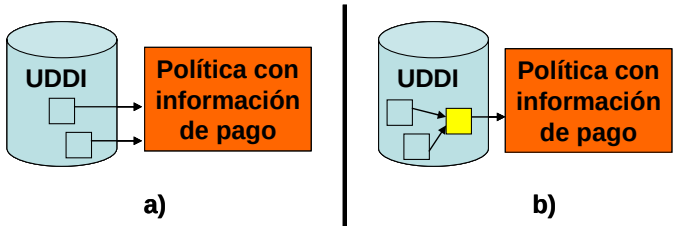
Figura 4.15. Información de pago en WSDL con WS-Policy.

4.3.4. Información de pago en UDDI

En esta sección describimos cómo podemos incluir la información de pago en UDDI. El objetivo es facilitar el descubrimiento de servicios y su información de pago. En UDDI existen dos posibilidades para incluir la información de pago.

La primera posibilidad es como un requerimiento utilizando WS-Policy y siguiendo el enfoque propuesto en la especificación *WS-Policy attachment* (adjunto en WS-Policy). La segunda posibilidad es incluyendo la información de pago directamente en UDDI de una forma similar a como se incluyen los WS-Policy Attachment en UDDI. A continuación explicamos ambas posibilidades.

La especificación WS-Policy attachment propone dos alternativas para incluir políticas (con o sin información de pago) en UDDI. En la primera opción, una entidad UDDI referencia directamente a una expresión de políticas remotamente accesible (ver Figura 4.16a). Este modelo se propone cuando una política es para un único servicio Web.



a) UDDI referencia a una política. b) tModel referencia a una política.

Figura 4.16. Información de pago en UDDI a través de WS-Policy.

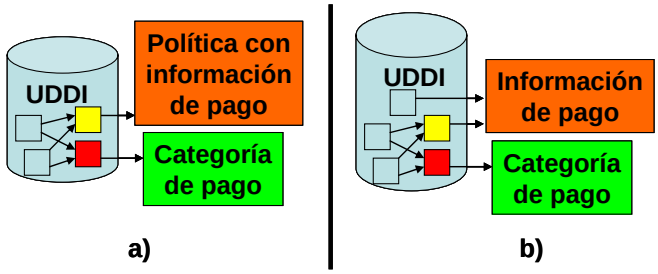
En el segundo enfoque, las entidades UDDI referencian a expresiones de políticas a través de *tModels* que registran expresiones de políticas (ver Figura 4.16b). Este enfoque es interesante porque es modular y permite la reutilización de políticas.

El principal problema de incluir la información de pago en WS-Policy es que no permite distinguir entre servicios que requieren pago y aquellos que no lo requieren. Para determinar si un pago es necesario, primero, tenemos que buscar los servicios que definen políticas y, a continuación, analizar que las políticas para determinar si el pago es necesario o no.

Para solucionar este problema, proponemos registrar cada elemento *PaymentInformation* con un *tModel* cuya categoría es *payment* (pago) (ver Figura 4.17a). Así, diferentes *businessServices* y *businessTemplates* pueden reutilizar la información de pago.

Para incluir la información de pago en UDDI sin WS-Policy definimos varios *tModels*. En primer lugar, un *tModel* para referenciar a un elemento *PaymentInformation* que es accesible remotamente. En segundo lugar, un *tModel* para acceder a los elementos *PaymentInformation* que está incluidos en el registro UDDI como un *tModel* y,

finalmente, necesitamos una categorización para definir que un *tModel* contiene información de pago (ver Figura 4.17b).



a) WS-Policy con categoría de pago. b) Información de pago sin WS-Policy.

Figura 4.17. Información de pago en UDDI.

4.3.5. Conclusiones

A lo largo de la Sección 4.3 hemos presentado cómo describir y adjuntar la información de pago en WSDL, WS-Policy y UDDI. En WSDL, hemos propuesto incluir la información de pago o extendiendo directamente WSDL o por medio de WS-Policy. También hemos definido cómo incluir esta información de pago en UDDI para facilitar la búsqueda, descubrimiento y composición de servicios Web. A esta propuesta la hemos denominado PA-WSDL. Además, hemos descrito cómo realizar el pago mediante EPP de forma que con estas dos propuestas podemos ofrecer un framework para servicios Web de pago. Las extensiones propuestas pueden ser utilizadas en conjunción con otras extensiones tales como SA-WSDL o WS-Agreement.

De nuestra propuesta se derivan varios beneficios potenciales:

- La información de pago está ligada a la descripción del servicio. Por tanto, el proceso de publicación y descubrimiento de servicios y su información de pago asociada se simplifica. Esto se debe al hecho de que bien a partir de la descripción WSDL o bien a partir de UDDI se puede encontrar la información de pago asociada.
- Se permite la negociación del precio del producto y las opciones de pago asociadas así como se permite elegir el protocolo de pago a utilizar cuando se combina con el protocolo EPP.
- El descubrimiento automático de servicios basado en agentes inteligentes [100, 101, 162] y UDDI [180] se facilita porque toda la información sobre el servicio se proporciona con su descripción y en un lenguaje que facilita su procesamiento como es XML. Por tanto, esta información podría ser utilizada junto con información semántica en agentes inteligentes para descubrir servicios que lleven a cabo operaciones expresadas de una forma semántica y que requieran un pago para ser utilizadas. Esta información también podría ser utilizada por los agentes para llevar

a cabo un proceso de comparación y negociación automático e inteligente de los precios y las características de los servicios.

- Es importante destacar que nuestra extensión puede ser complementaria a otras extensiones de WSDL, tales como P3P para privacidad [64], SA-WSDL [159] que facilita la búsqueda de servicios basada en esos agentes automatizados que acabamos de mencionar o WS-Agreement para negociar acuerdos a nivel de servicio (Service Level Agreements – SLAs) entre el proveedor del servicio y el consumidor de éste.
- Esta solución proporciona una manera uniforme de publicar servicios Web de pago. Este hecho puede motivar a los proveedores de servicios a proporcionar las descripciones de sus servicios en servidores UDDI o en cualquier otro directorio de servicios. Así se facilita a los clientes determinar, de una forma sencilla, la mejor oferta dependiendo de sus requerimientos. De esta manera se promueve la competencia y el desarrollo de servicios que ofrezcan una mejor calidad.

4.4. Conclusiones del capítulo

En el Capítulo 2 presentamos los frameworks de pago como una respuesta a la necesidad de soportar el uso de distintos sistemas de una forma uniforme, que garanticen la confianza del usuario en los sistemas de pago y que faciliten a vendedores, bancos y desarrolladores de software de pago la incorporación de estos sistemas de pago. Así los clientes pueden utilizar el sistema de pago que crean más conveniente y, por otro lado, los vendedores pueden ofrecer distintos mecanismos de pagos a sus clientes de forma que ofrezcan un mejor servicio a los clientes. Sin embargo, como se comentó ninguna de ellas parecía ofrecer una solución global a la necesidad de frameworks de pago que nos permitirán fácilmente la compra de contenidos y servicios Web.

En respuesta a esta necesidad, en este capítulo, en primer lugar hemos propuesto un framework de pagos por clic que es independiente del protocolo de pagos, pero que está diseñado especialmente para soportar micropagos ya que éstos son la clase de sistema ideal de pago para contenido de bajo valor. Este framework está basado en el diseño de varios componentes que han sido definidos de forma genérica con el fin de que se puedan soportar cualquier sistema de pago así como se puedan utilizar en otros escenarios o frameworks. En concreto, estos elementos son: un protocolo de pagos genérico denominado Extended Payment Protocol (EPP), cómo expresar la información de pago por medio de XML y, finalmente, el diseño de un wallet que permite abstraer de las características específicas de los protocolos de pagos al framework. Este framework, además ha sido desarrollado y evaluado con dos protocolos de pago y se ha mostrado que su uso es viable y que la sobrecarga introducida podría ser aceptable por los usuarios a la hora de realizar pagos.

El framework que acabamos de mencionar, como decíamos, está orientado al modelo de pago por clic para el acceso, principalmente, a contenidos Web. Sin embargo, como explicamos, su uso no sería muy adecuado para el pago por servicios Web. Para este escenario, hemos diseñado un framework basándonos en los elementos definidos para el

framework de pagos por clic que antes hemos mencionado. Además de estos componentes, definimos como incorporar la información de pago en WSDL y UDDI, que es lo que hemos denominado *Payment Annotations for WSDL* (PA-WSDL).

En la Figura 4.18 se muestra una visión conceptual de los distintos elementos que forman parte de los frameworks propuestos. En el nivel inferior podemos observar los distintos protocolos de pagos. A continuación, en el siguiente nivel podemos observar los wallets de pago que encapsulan estos protocolos. Estos wallets serán utilizados por el protocolo de pago EPP para la adquisición de contenidos y servicios Web. En el nivel superior podemos encontrar las distintas especificaciones que formarían parte del nivel de aplicación o servicios para los distintos escenarios sobre los que utilizaremos EPP. Finalmente, podemos observar que hay una capa transversal contiene los elementos necesarios para expresar la información de pago y que puede ser utilizada en distintos niveles.

Por tanto, en vista de estos resultados podemos concluir que con estas soluciones hemos definidos dos frameworks que permiten la adquisición de contenidos y servicios Web mediante una serie de elementos genéricos que facilitan el pago con distintos protocolos de pago. Estos elementos genéricos son los que fomentan el desarrollo de ambos escenarios de pago ya que se pueden reutilizar los componentes desarrollados. Así la implementación de cualquier protocolo mediante un wallet podrá utilizarse en ambos escenarios sin problemas. Además, estas soluciones facilitan la búsqueda y el descubrimiento de información de pago en la Web y la automatización de la negociación de los precios y protocolos a utilizar.

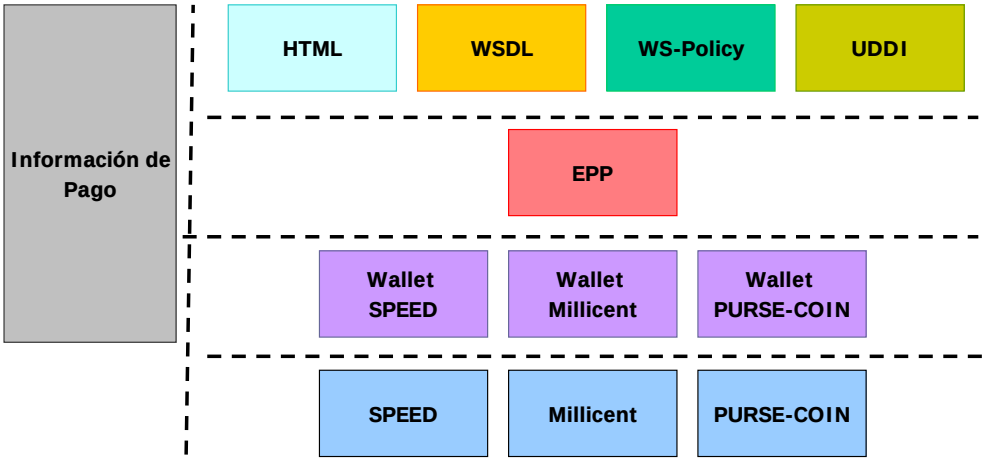


Figura 4.18. Vista conceptual de los componentes de los frameworks.

Capítulo 5

Frameworks de pagos para el acceso a servicios multimedia

En el capítulo anterior presentamos un framework genérico para el soporte de pagos por clic. Dicho framework está pensado para realizar la compra de productos que se distribuyen vía Web ya que, a día de hoy, es uno de los principales mecanismos de distribución de contenidos o productos electrónicos. A pesar de estar centrado en un entorno en particular, los componentes diseñados y desarrollados para describir la información de pago o soportar un determinado protocolo podrían ser utilizados en otros escenarios debido a su diseño genérico, tal y como también presentamos para el framework de pagos para servicios Web.

En este capítulo abordamos cómo realizar la adquisición y el pago de productos multimedia que se distribuyen bajo demanda (por ejemplo, vídeo bajo demanda) y servicios basados en el establecimiento de sesiones (multimedia). Para este tipo de productos se podrían utilizan los frameworks propuestos en el capítulo anterior. Sin embargo, debido a las particularidades de la distribución de estos tipos de productos es más adecuado realizar el proceso de pago por medio de los protocolos de control utilizados para éstos. Estos protocolos se utilizan para establecer una sesión para la distribución del contenido o del servicio. Además, en ellos también realiza la negociación de las características del servicio a proveer. Al utilizar el intercambio de información llevado a cabo en estos procesos evitamos tener que utilizar un protocolo adicional para realizar este proceso.

A pesar de no utilizar los frameworks anteriormente definidos si que utilizaremos algunos de los elementos que éstos proporcionan. Así, para facilitar el pago de cada uno de estos tipos de productos o servicios electrónicos multimedia hemos seguido los mismos principios de diseño del capítulo anterior. Básicamente, la idea será poder expresar la información de pago de forma genérica, soportando el uso de distintos protocolos de pago, marcas financieras, proveedores de servicios de pago, precios, la negociación del precio del producto y el uso de cualquier protocolo de pago para realizar la compra del producto deseado. Además de seguir los mismos principios, se mostrará cómo los anteriores

componentes efectivamente son lo suficientemente genéricos para ser utilizados en otros escenarios.

5.1. Introducción

El hecho de que cada vez haya más ancho de banda de acceso y mayor número de usuarios conectados a Internet, motiva y permite a los vendedores o proveedores de contenidos ofrecer nuevos productos o servicios electrónicos que años atrás no eran posibles debido a la falta de ancho de banda disponible en el lado del cliente. En concreto nos estamos refiriendo a servicios como la distribución de contenidos bajo demanda y el acceso a servicios multimedia que se sirven a través de la definición de sesiones.

El acceso a estos contenidos o servicios puede ser gratuito, basado en suscripción, facturación o realizando un pago por cada servicio o producto particular. En la siguiente sección realizaremos un análisis del uso de cada uno de estos tipos de acceso y de las preferencias de los clientes y los vendedores. Veremos que en determinadas circunstancias el pago por el acceso por cada producto o servicio particular es una opción interesante para clientes y vendedores. A día de hoy, las soluciones ofrecidas para llevar a cabo el pago de los servicios anteriormente mencionados principalmente están basadas en sistemas de suscripción o facturación. También existen soluciones que soportan la realización de pagos, aunque las soluciones presentadas hasta la fecha están basadas en soluciones propietarias o limitadas a un mecanismo de pago en particular, como veremos posteriormente.

En el Capítulo 2 analizamos las distintas soluciones de pago que tratan de cubrir el pago de una forma genérica. Posteriormente en el Capítulo 4, presentamos una solución que está orientada al pago de enlaces de una página Web y, basándonos en ésta, también propusimos una solución para el pago por el uso de servicios Web. También se podría utilizar para el pago de contenidos multimedia distribuidos bajo demanda o mediante sesiones. Sin embargo, para estos últimos tipos de contenidos, como explicaremos en este capítulo, será más adecuado seguir otro enfoque. En este caso será la extensión de los protocolos para acceder a dichos servicios. El objetivo será poder negociar no sólo el precio del servicio sino también las características con las que se distribuye éste (codecs, calidades, etc) y realizar el pago a la misma vez que se lleva a cabo la inicialización o el acceso a tales contenidos. De esta manera tratamos de evitar el intercambio de mensajes adicionales así como de múltiples conexiones entre las mismas entidades.

En respuesta a esta problemática, este capítulo tiene como primer objetivo ofrecer soluciones que permitan la adquisición de contenidos o servicios electrónicos multimedia que se distribuyen bajo demanda o por medio de sesiones. De esta forma, las soluciones ofrecidas aquí junto con las soluciones planteadas en el Capítulo 4 nos permitirían cubrir los principales escenarios para la adquisición de productos o servicios electrónicos en el comercio B2C.

El segundo de nuestros objetivos será que las soluciones para adquirir estos contenidos sigan los mismos principios planteados en el capítulo anterior. Es decir, las soluciones serán genéricas y soportarán:

- distintos mecanismos de pago.
- diferentes modelos de negocio.
- negociación de las opciones de pago y del precio del producto.
- encapsulación del envío de los mensajes de pago del protocolo de pago elegido.

Otro de nuestros objetivos es que el diseño de las soluciones esté basado en los componentes presentados en el Capítulo 4 para la descripción de la información de pago y la gestión de los métodos de pago.

Las soluciones además tratarán de estar basadas en los distintos estándares y propuestas disponibles para la distribución de tales contenidos o servicios con el fin de que la solución pueda ser ampliamente utilizada. En caso de que estas soluciones no satisfagan los requisitos establecidos buscaremos incorporar estas características en base a los mecanismos de extensibilidad que éstas ofrecen. Nuestra meta será aprovechar los mecanismos de control que puedan soportar estas soluciones para evitar tener que intercambiar mensajes adicionales. De esta forma evitamos mayor sobrecarga en las comunicaciones así como la sincronización entre distintos elementos. Si el proceso de pago estuviera desligado del proceso de reproducción de todas formas sería necesario crear algún mecanismo de autorización (léase recibo o token) para obtener el acceso a la información. Para este caso también sería necesario extender los protocolos mencionados.

A continuación, en la Sección 5.2, comentaremos los antecedentes relacionados con el acceso y pago de los servicios multimedia descritos anteriormente. Una vez introducimos los antecedentes pasaremos a introducir nuestras propuestas. Estas propuestas se basan en la extensión de los estándares SDP (Session Description Protocol – Protocolo de Descripción de Sesión) [123], SIP (Session Initiation Protocol – Protocolo de Inicio de Sesión) [240] y RTSP (Real-Time Streaming Protocol – Protocolo de Streaming en Tiempo Real) [270] para soportar el pago por servicios que están basados en el establecimiento de una sesión con SIP y por contenidos que se distribuyen bajo demanda con RTSP.

5.2. Antecedentes

En esta sección realizamos un análisis, desde el punto de vista de negocio, de las opciones que existen para realizar el acceso a servicios multimedia distribuidos bajo demanda o bajo sesiones. Los analizamos de forma conjunta porque, aunque hay propuestas distintas para cada uno de ellos, hay determinados mecanismos de pago que podrían ser comunes como son la suscripción o la facturación. En este análisis prestaremos especial atención al modelo de negocio basado en pagos individualizados para el acceso a estos servicios.

5.2.1. Modelos de negocio electrónico para servicios multimedia

En general, en los modelos de negocio electrónico y en particular para aquellos que involucran la distribución de servicios multimedia, el acceso gratuito se ofrece a los

usuarios para que puedan probar el servicio. Si a los usuarios finalmente les gusta el servicio, se les invitará a que paguen una suscripción o a efectuar pagos por contenidos adicionales. Algunos proveedores de contenidos o servicios incluso sólo ofrecen contenidos de forma gratuita. En este caso, los proveedores obtienen el beneficio a partir de la publicidad que insertan en los contenidos o servicios prestados [229].

Desde el punto de vista del beneficio del vendedor hay dos principales modos de acceso a servicios multimedia: aquellos que están basados en suscripción y aquellos que se basan en un pago individual por un determinado contenido. Este pago se puede realizar justo en el momento antes de consumir el producto o servicio (pay-per-view), pagando conforme se recibe el contenido (pay-as-you-watch) o al final de un determinado período de tiempo (semana, mes, año) mediante el pago de la factura que el cliente tiene con su proveedor de servicios a Internet (normalmente mediante un cargo en una cuenta bancaria del cliente).

El pago de estos tipos de servicios se puede llevar a cabo de distintas formas: mediante el pago de una suscripción, por medio de tarificación (facturación) o mediante la realización de un pago por cada contenido o servicio al que se accede (modelos pay-per-view o pay-as-you-watch).

En general, el modelo basado en suscripción es el preferido por los vendedores. Esta preferencia se debe a que obtienen beneficios durante el período de suscripción independientemente del acceso del cliente. Además, tienen garantizado un determinado número de clientes durante dicho período.

En este modelo los usuarios realizan un pago por adelantado sobre el uso del servicio durante un determinado período de tiempo (semana, mes o año). Así, los vendedores obtienen el pago por adelantado y de forma independiente al uso que posteriormente el usuario lleve a cabo del servicio (o de los contenidos a los que acceda). Este método se puede aplicar a los distintos tipos de productos o servicios que hemos mencionado anteriormente. Con este sistema, si el usuario decide no utilizar más el servicio durante el período de suscripción, el usuario está perdiendo parte de su dinero [75, 76, 184]. Otro problema de los sistemas basados en suscripción es el anonimato ya que el sistema puede conocer qué productos o servicios consume cada suscriptor así como las relaciones que existen entre las distintas entidades.

Algunos proveedores de contenidos incluso combinan la suscripción con el pago por uso. Este enfoque mixto lo ofrecen como un elemento diferenciador frente a otros vendedores con el objetivo de que les permita obtener una mayor fidelización de sus usuarios [173]. La suscripción garantiza el acceso a servicios que el usuario utiliza habitualmente. Adicionalmente, se pueden solicitar pagos por servicios especiales o particulares que el usuario utiliza ocasionalmente y que no están incluidos en la suscripción [229].

Otra de las posibilidades que existen para realizar el pago de estos servicios (multimedia) es mediante la factura que recibimos de nuestro proveedor de servicios a Internet (ISP) o de nuestro operador de red. En este caso, el proveedor del servicio multimedia llega a un acuerdo con el proveedor de acceso a la red del cliente para que sea éste el que cobre el importe asociado al uso del servicio (o de los contenidos accedidos). Así, el proveedor de acceso a la red cobra en la factura semanal/mensual/anual el importe de sus servicios junto con los servicios prestados por los distintos proveedores de servicios

multimedia que el usuario ha utilizado durante el período por el que se factura. Este sistema se puede aplicar a los productos o servicios mencionados.

El modo más extendido para llevar a cabo la tarificación de los servicios de red o contenidos electrónicos consumidos es por medio de los protocolos AAA (*Authentication, Authorization and Accounting* – Autenticación, Autorización y Auditoría) [71]. La principal ventaja es que los clientes no tienen que realizar ningún proceso de envío de información de pago a los proveedores del servicio (multimedia), así como tampoco requiere proceso de registro con éstos, sólo es necesario un proceso de autenticación/autorización en el sistema (en general, con el servidor AAA del proveedor de acceso a la red). Sin embargo, esta solución tiene dos inconvenientes fundamentales, en primer lugar, el proveedor del servicio (multimedia) tiene que enviarle la información de auditoría (accounting) al proveedor de acceso a la red (convirtiéndose en un cliente AAA), con lo que éste podría crear perfiles de los usuarios y los servicios que utilizan (dependerá en gran medida de qué tipo información se envía en las trazas). Segundo, el usuario sólo puede acceder a aquellos servicios con los que su proveedor de acceso a la red tiene establecidos acuerdos. Finalmente, en general, a los usuarios no les transmite confianza que el cobro de los servicios venga junto con la factura de su proveedor de acceso a la red [227].

Por otro lado, los clientes, en general, sólo prefieren pagar por los contenidos que realmente solicitan y prefieren no recibir publicidad [229]. Además, por norma general, prefieren realizar el pago de los productos o servicios en el momento que tiene lugar su adquisición en lugar de que el pago se produzca posteriormente en la factura que se adjunta con los servicios proporcionados por su ISP o su proveedor de servicios de red [227]. También hay que tener en cuenta que muchos de los contenidos multimedia están orientados a los jóvenes que, habitualmente, no suelen tener ni cuenta bancaria ni tarjeta de crédito. Sin embargo, éstos sí que podrían poseer dinero electrónico en forma de monedas electrónicas o en forma de tarjeta prepago.

En cuanto a la cantidad a pagar, en general, los servicios basados en suscripción requieren pagos de mayores cantidades que los modelos de pay-per-view porque los servicios son más amplios o la duración en el tiempo es mayor. Los sistemas de pay-per-view o pago por uso, normalmente, requieren el soporte de pagos de medio-bajo valor. Para estos pagos, como mencionamos en el Capítulo 2, se recomienda el uso de micropagos o monedas electrónicas debido a la relación coste-efectividad en términos de seguridad, sobrecarga de los protocolos y la información a procesar en el proceso de validación de un pago concreto. Este modelo de pago lo analizamos en la siguiente sección.

5.2.2. Modelo de pagos para el acceso servicios multimedia

Para llevar a cabo el pago cada vez que se realiza el acceso a un servicio multimedia o Web se han seguido dos enfoques. El primero de ellos es el desarrollo de soluciones particulares centradas en un determinado tipo de software y con un determinado protocolo de pago. La otra opción se trata de ofrecer soluciones que sean lo suficientemente genéricas para que un amplio número de vendedores puedan utilizarlas.

Para algunos de los distintos tipos de productos que hemos mencionado existen algunas propuestas basadas en alguno de estos enfoques. Estas propuestas las comentamos a continuación. Las propuestas más interesantes serán descritas posteriormente más en detalle.

En el caso de soluciones particulares para contenidos multimedia distribuidos bajo demanda tenemos el caso del proyecto STREAMOBILE [75, 76]. En este proyecto se presenta una arquitectura para vídeos de pago transmitidos bajo demanda en la que el pago se efectúa cada vez que se ha recibido parte del contenido (pay-as-you-watch). En concreto en esta propuesta el método de pago elegido es Payword [238] (se puede encontrar información adicional sobre Payword en el Capítulo 2).

Además, existen otras propuestas como la de la empresa Helix que ha desarrollado sus propios mecanismos para recibir pagos. Esta propuesta permite al usuario elegir entre los distintos mecanismos de pago soportados así como es extensible.

El principal problema de ambas soluciones es que limitan la extensibilidad de la solución. El proveedor del servicio tiene que desarrollar (o instalar de un tercero) los componentes de servidor y cliente que soporten esta solución concreta. El número de sistemas de pago queda limitado a los que desarrolló el proveedor del servicio (o al tercero al que se adquirió la solución). Además, el cliente puede que tenga que cambiar el reproductor que habitualmente usa para utilizar el desarrollado para la solución en particular ya que no están basados en mecanismos de extensibilidad de los protocolos ya existentes. Este hecho también provoca que las soluciones entre los distintos vendedores no puedan interoperar, ni el software de los distintos mecanismos de pago tampoco puede ser utilizado con el software de los distintos clientes. Finalmente, mencionar que estas soluciones no permiten la negociación de los precios asociados a las opciones de pago.

En el ámbito de las soluciones génericas para realizar pagos tenemos varias posibilidades. Una es la utilización del protocolo IOTP [32, 33] (comentado en el Capítulo 2) o el protocolo EPP [253] propuesto en el Capítulo 4. Ambas soluciones se podrían utilizar para pagar el tipo de contenidos que hemos mencionado.

El problema de ambas soluciones viene dado por el hecho de que, en ambos casos, el proceso de pago está desligado de la solicitud de acceso. Este hecho implica dotar de algún mecanismo (habitualmente mediante la extensión de los protocolos de acceso) para permitir que, una vez realizado el pago, se permita acceder al contenido. En concreto había que definir cómo se envía la información de pago y cómo es posible ligar ambos procesos. Además, para el caso de productos digitales bajo demanda o distribuidos bajo sesión no permiten negociación de las características de la sesión, además de requerir conexiones adicionales entre las distintas partes. Dependiendo de las características de la sesión (codecs, calidad, etc), los precios serán distintos. Por tanto, en el proceso de pago habría que realizar un proceso de negociación no sólo del precio si no también de la información multimedia a intercambiar. Este intercambio también se tiene que producir en el protocolo de acceso, con lo cuál estaría duplicando el proceso e intercambiando información innecesaria.

Dentro de las soluciones genéricas, en concreto para SIP, Jennings et al. [146] propusieron una solución que está basada en la integración con una tercera entidad. Esta

solución propone realizar el pago, en vez de al vendedor, a una tercera parte que actúa como Proveedor de Servicios de Pago (PSP) llevando a cabo una transferencia desde la cuenta del cliente a la del vendedor. El principal problema de esta propuesta es que no permite la utilización de distintos protocolos de pago y, por tanto, no se podría utilizar protocolos que garantizasen el anonimato o el uso de monederos. Además, el problema de la participación de esta PSP es que podría no ser adecuada para micropagos como comentamos en el Capítulo 2. Otro problema asociado es que podría conocer ciertos detalles de la transacción así como las partes involucradas. Un análisis más detallado de esta solución se presenta a continuación.

5.2.3. Propuesta de Jennings et. al.

Durante los años 2004 a 2008 Jennings et al. [146] han estado trabajando en un draft de Internet para permitir que una entidad que recibe una llamada o solicitud de servicio en SIP pueda cobrar al que realiza dicha llamada o solicitud de servicio.

En el modelo que proponen participan tres entidades: el consumidor, el vendedor y el proveedor de pagos. Este escenario está basado en el uso de SAML (Security Assertion Markup Language – Lenguaje de Marcas de Sentencias de Seguridad) y cada una de estas partes se corresponde con los siguientes roles SAML: solicitante SAML, respondedor SAML (entidad del sistema que utiliza el protocolo SAML para responder a una solicitud de servicios de otra entidad del sistema) y autoridad SAML, respectivamente.

El proceso de pago es el que se muestra en la Figura 5.1. En este proceso el cliente intenta acceder a un servicio del vendedor. Éste le indica que el servicio requiere un pago y las características del pago (precios y proveedores de pago soportados – proveedores de servicios de pago conforme a la terminología que hemos utilizado en los otros capítulos). A continuación, el cliente realiza un pago a un proveedor de servicios de pago (PSP).

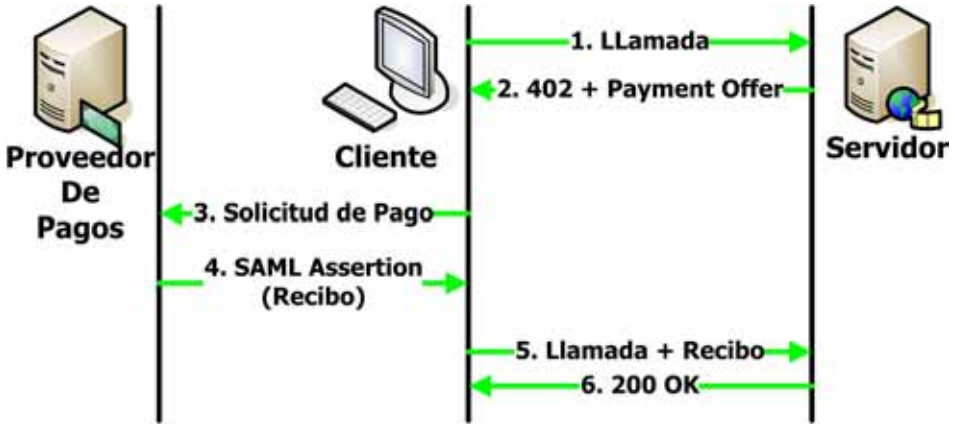


Figura 5.1. Modelo de pago en SIP de acuerdo a la propuesta de Jennings et al.

Una vez que el pago se ha realizado, el PSP, como confirmación de pago, genera un recibo de pago basado en SAML. Este recibo se utiliza posteriormente por el cliente para acceder al servicio del vendedor.

Esta propuesta introduce algunas características que son interesantes:

- El cliente puede elegir su marca financiera preferida al mismo tiempo que el vendedor es independiente de esta elección y no tiene que soportar ningún protocolo de pago en particular.
- El vendedor recibe, en lugar de un pago, un recibo integrado en la ejecución de SIP.
- El vendedor puede ofrecer a los clientes diferentes modelos de pago tales como tarifa plana, por unidad de tiempo y por unidad de datos. También puede soportar múltiples divisas y diferentes PSPs.
- El cliente puede extender una sesión enviando un nuevo recibo de pago.

A pesar de estas características, esta propuesta también presenta varias limitaciones. Aunque la participación de una tercera entidad (PSP) para llevar a cabo el proceso de pago es interesante para mantener la independencia del vendedor con respecto al sistema de pago, supone que el PSP tiene que participar en cada pago. Esto puede ser útil cuando el importe a pagar se corresponde con una transacción de valor medio-alto ya que esta entidad proporciona confianza en el sistema. Sin embargo, como se mencionó en el Capítulo 2 y en [117, 139, 160], la participación de una tercera entidad no es adecuada para transacción de bajo valor o micropagos ya que podría suponer altos costes de transacción. En estos sistemas, la eficiencia y los bajos costes son las características fundamentales a soportar frente a otras como puede ser el intercambio equitativo o la posibilidad de pérdidas. El uso de (micro)pagos en SIP se ha propuesto para diferentes escenarios [103, 146] como, por ejemplo, como una solución al problema del SPAM en los sistemas VoIP, el acceso a servicios multimedia que pueden ser facturados con distintos modelos de negocio: por minuto, por hora o tarifa plana; servicios de comunicación en tiempo real en entornos P2P, etc.

Otro problema relacionado con la participación de esta entidad (que podría ser un banco, una pasarela de pago, un proveedor de pagos, etc) es que puede conocer todos los detalles de la transacción (por ejemplo, identidad de las partes, cantidades transferidas, frecuencia, etc). Así, esta propuesta impide que los clientes puedan utilizar protocolos de pago que garanticen características como el anonimato o el uso de monedas o protocolos de pago que usen tarjetas inteligentes que proporcionan un nivel de seguridad adicional, etc. Además, si la relación entre el cliente y el vendedor es esporádica y la cantidad es de bajo valor, la transferencia entre bancos podría ser más costosa que utilizar un protocolo específico para este tipo de transacciones. Por tanto, esta solución no es flexible y puede ser adaptada diferentes requisitos de seguridad y/o funcionalidad.

Otra limitación importante es el hecho de que no tiene en cuenta que los flujos de datos con diferente calidad pueden tener precios diferentes a pagar en una sesión. Finalmente, esta propuesta no permite la especificación de diferentes cantidades a pagar con los diferentes PSP a utilizar. Deberíamos de tener en cuenta que cada proveedor de pagos podría cobrar diferentes comisiones o tasas a un vendedor a cambio de sus servicios.

5.2.4. STREAMOBILE

STREAMOBILE [75, 76, 184] es un proyecto desarrollado en 2002 que tenía por objetivo mostrar la distribución vídeo bajo demanda sobre dispositivos móviles. En este sistema los derechos de copyright de los contenidos distribuidos se protegen mediante técnicas de watermarking y fingerprinting.

Esta propuesta también es interesante porque introduce el concepto de pay-as-you-watch. Así, en vez de que el contenido se pague antes de que el usuario lo haya visto, el usuario paga cuando está recibiendo el contenido emitido bajo demanda. Es decir, el usuario sólo paga por la parte de los contenidos que ha visto y puede finalizar la reproducción en cualquier momento. De esta forma, como el contenido se paga conforme se recibe, los pagos se van realizando sobre pequeñas cantidades y por este motivo como mecanismo de pago adoptaron Payword [238].

Para el pago la tienda y el wallet del cliente establecen una sesión para que el vendedor pueda realizar las sucesivas solicitudes de pago al cliente, en las que éste enviará cada vez una moneda Payword. Una vez que el pago se ha realizado el servidor de contenidos continúa enviando n Kbytes del vídeo al reproductor de contenidos.

Esta solución, como se puede deducir, no está pensada para ser utilizada con distintos protocolos de pago. Por tanto, tampoco soporta la negociación de distintos protocolos así como no soporta la negociación de la calidad del contenido.

En esta solución la distribución de contenido se lleva a cabo mediante HTTP, por lo que no ofrece el mismo control de flujo de datos que se ofrece con otros protocolos más adecuados como pueden ser RTSP y RTP [271]/SRTP [17]/ZRTP [325].

5.2.5. Conclusiones

A lo largo de esta sección de antecedentes hemos analizado las distintas soluciones que a día de hoy existen para la realización del pago de productos o servicios electrónicos multimedia que se distribuyen bajo demanda o bajo sesiones.

Hemos presentando soluciones de pago particulares para los productos o servicios bajo demanda o basados en sesiones. Sus principales limitaciones son que los mecanismos de pago utilizados son reducidos y además, las soluciones diseñadas basadas en este tipo de propuestas dificultan la extensibilidad de las soluciones existentes, reduciendo el ámbito en que se podrían utilizar este tipo de soluciones.

También hemos mencionado frameworks de pagos genéricos que podrían ser utilizados para realizar este tipo de pagos permitiendo al usuario elegir entre distintos mecanismos de pago. Sin embargo, este tipo de soluciones, para estos tipos de productos requerirían el establecimiento de conexiones adicionales así como desligar el proceso de pago del proceso de adquisición del producto o servicio.

5.3. Soporte de pagos en SIP

5.3.1. Introducción

El principal objetivo de la propuesta que presentamos en esta sección es el soporte del pago de sesiones multimedia en SIP (Session Initiation Protocol – Protocolo de Inicio de Sesión) [240] de una forma genérica conforme a los objetivos planteados al principio de este capítulo.

Esta solución tiene por objetivo proporcionar una serie de características que describimos a continuación:

- Permitir describir sesiones multimedia indicando la información de pago asociada. Además, debe permitir que se especifiquen distintos precios para flujos de datos con distinta calidad así como distintos precios para las distintas opciones de pago que se ofrezcan.
- El framework debe soportar el pago con cualquier protocolo de pagos (micropagos o macropagos). Incluso deber ser posible soportar pagos basados en una tercera parte que genere recibos de pago como, por ejemplo, proponen Jennings et al [146]. La información para describir la información de pago y los componentes que implementan los protocolos de pago serán los diseñados en el Capítulo 4.
- Soportar la negociación del precio y de las opciones de pago.
- Permitir que el vendedor pueda adjuntar recibos de pago para el cliente. Este recibo no tiene por objetivo servir para el mismo propósito que en la propuesta de Jennings et al. En su propuesta el recibo se utiliza para proporcionar acceso a la sesión multimedia y, también, puede ser utilizado como prueba de compra. En nuestra propuesta, el recibo que se proporciona sólo sirve como prueba de pago. Aunque no se descarta que pudiera ser utilizado para posteriores accesos con el mismo propósito que con la solución anteriormente mencionada.
- Soportar la realización de pagos incrementales o adicionales, es decir, un cliente, en vez de realizar el pago por la sesión completa, podría pagar por una parte de la sesión. La sesión comenzaría y, más tarde, antes de que la reproducción pagada llegue a su fin, si el usuario lo desea, podría realizar un pago adicional para continuar con la recepción de los contenidos multimedia de la sesión. También se podría soportar el modelo pay-as-you-watch. De forma que se inicie la sesión sin pago alguno y, posteriormente, se vayan solicitando estos pagos.
- Estar basado en los mecanismos de extensibilidad que ofrece SIP [182, 185] para facilitar su aceptación y la incorporación a las soluciones ya existentes.

A continuación, en la siguiente sección proporcionamos una visión genérica de cómo hemos extendido SIP para soportar las características que acabamos de mencionar. Posteriormente, explicaremos en detalle los distintos componentes en los que esta propuesta está basada.

5.3.2. Visión genérica del framework de pagos en SIP

Cuando un usuario está interesado en acceder a un servicio multimedia de pago por medio de SIP, éste podría obtener información acerca de la sesión antes de su inicio o intercambiarla durante el inicio de la sesión y acto seguido realizar el pago.

La información que el cliente podría obtener antes de iniciar la sesión sería la información que describe la sesión (flujos, codecs, calidades, etc). También podría obtener las opciones de pago soportadas y el coste asociado. Si el cliente obtiene esta información por adelantado se puede simplificar la información a intercambiar durante el inicio de sesión SIP.

Para acceder al servicio será necesario realizar el proceso de inicio de sesión de SIP en el que se realizará el intercambio de la información que describe la sesión y la información de pago asociada. Con este intercambio el cliente puede obtener las características del servicio (codecs, calidades, etc). Además, podría negociar el precio de estas características. El precio de éstas podría depender de los codecs y calidades así como del protocolo utilizado para realizar el pago. El precio puede estar expresado de varias formas: como una cantidad por el servicio o en unidades de tiempo (segundos, minutos, horas) o por cantidad de datos. De estos modelos destaca el modelo de tiempo ya que de esta manera, los usuarios pueden saber la cantidad a pagar por el periodo de tiempo que quieren recibir.

Sin embargo, el cliente podría no estar de acuerdo con el importe que el vendedor le solicita y podría querer negociar el precio de este contenido. Si el vendedor soporta esta posibilidad y está dispuesto a negociar el precio, tendría lugar una fase de negociación donde se podría negociar la calidad y el precio asociado.

Cuando el cliente finalmente está de acuerdo con las condiciones (opciones de pago y precio), realiza un pago por el tiempo deseado. Cuando el proveedor del servicio comprueba que la información de pago es válida, éste empieza a proveer el servicio solicitado. Durante el pago el cliente podría proporcionar información de fidelización que le permitan obtener un mejor precio o determinadas credenciales que le permitan acceder al servicio. Una vez realizado el pago el vendedor también podría proporcionar nueva información de fidelización así como justificantes de pago de la transacción realizada.

Una vez que el tiempo por el que usuario pagó finaliza, el proveedor finaliza la sesión. El comportamiento descrito hasta ahora es lo que denominamos el modo básico.

En el modo avanzado, el framework proporciona una característica importante: el soporte de pagos adicionales sin la necesidad de parar la provisión del servicio.

En el framework, el servidor puede avisar al cliente de que el tiempo por el que pagó está llegando a su fin. Si al cliente le gusta el servicio, podría estar dispuesto a realizar un pago por más tiempo. Así, la reproducción continuaría sin ninguna interrupción ya que el pago se efectúa en paralelo con la provisión del servicio.

A continuación, en los siguientes apartados explicaremos cómo se realizan cada uno de los procesos que acabamos de explicar. En primer lugar, explicaremos cómo obtener las opciones de pago soportadas antes de iniciar la sesión. Después, describiremos el proceso de inicio de sesión y el mecanismo de pago básico. Finalmente, explicaremos el resto de

características adicionales que éste soporta, como es la negociación del precio, el envío de recibos e información de fidelización y los pagos adicionales.

Comprobación de opciones de pago soportadas

En determinadas circunstancias, podríamos estar interesados en que las diferentes opciones de pago soportadas por el vendedor pudieran ser conocidas por adelantado por los clientes. Así, el número de opciones de pago que el cliente enviaría durante el intercambio que se produce en el inicio de sesión en el que se produce el pago sería más reducido.

En el framework hemos hecho posible este intercambio por medio de la extensión del método estándar *OPTIONS* y de su respuesta. Ambos han sido extendidos para que la estructura *PaymentInformation* pueda ser enviada en el cuerpo del mensaje. Esta es una forma simple, estándar y eficiente de llevar a cabo este proceso de consulta.

Mecanismo de pago básico

En general, el proceso de pago consiste en una serie de pasos que describimos a continuación. En primer lugar, el vendedor proporciona las opciones de pago que soporta. El cliente elije una de esas opciones de pago y, a continuación, el proceso de pago comienza. Este proceso, en general, supone el intercambio de varios mensajes de pago entre el cliente y el vendedor. La mayoría de los sistemas de micropagos y protocolos de dinero electrónico sólo requieren dos mensajes, dependiendo de si el protocolo genera monedas por el importe exacto a pagar o no. Sin embargo, los protocolos de macropagos podrían requerir el intercambio de más mensajes como, por ejemplo, en SET. A continuación, describimos como intercambiar esta información de pago en SIP a la misma vez que se inicia la sesión.

Una sesión SIP se establece entre los participantes por medio un protocolo de tres vías. En esta sesión, un participante desempeña el rol del cliente y el otro actúa como servidor. El rol de cliente es representando por aquél que inicia la sesión.

La sesión se inicia con el mensaje *INVITE* y se describe por medio de SDP, que enumera los medios que la componen. Cuando la solicitud de *INVITE* se recibe, el vendedor retorna un código de respuesta *200 (OK)* y el cliente finaliza el protocolo con un mensaje *ACK*.

Pago exacto basado en moneda o recibo

El intercambio que acabamos de mencionar nos permite mapear directamente el pago con un protocolo basado en moneda electrónica (siempre que la moneda generada sea por el importe exacto a pagar) o con un sistema de pago basado en una tercera parte que genera un recibo de la transacción como token de acceso.

Con el fin de realizar el pago a la misma vez que iniciamos la sesión, hemos extendido los métodos que acabamos de mencionar (ver Figura 5.2). Así, el cliente primero envía el mensaje *INVITE* conteniendo una descripción SDP en el cuerpo del mensaje. Con esta descripción, opcionalmente, el cliente podría enviar las opciones de pago que soporta. Para

intercambiar esta información nos basaremos en la extensión de SDP que proponemos en la Sección 5.3.3.

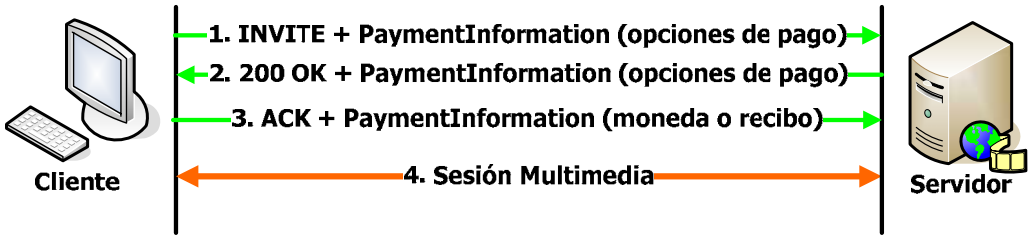


Figura 5.2. Pago con moneda electrónica o recibo.

El vendedor responde con el código de respuesta *200 (OK)*. Este mensaje en el cuerpo contiene un SDP extendido que representa la solicitud de pago indicando las opciones de pago soportadas y los precios para el servicio solicitado por el cliente. El conjunto de opciones de pago enviadas por el vendedor es un subconjunto de las opciones del cliente que el vendedor soporta.

Finalmente, en el mensaje *ACK* el cliente enviaría un recibo o la moneda necesaria para realizar el pago requerido.

La información para realizar el pago así como la información de pago contenida en el SDP extendido se envía por medio de la estructura *PaymentInformation* descrita en la Sección 5.3.3. Esta estructura la utilizaremos en esta propuesta en diferentes situaciones que serán comentadas a lo largo de esta sección. Con este intercambio, el pago puede ser llevado a cabo a la misma vez que se establece la sesión multimedia sin mensajes adicionales.

Pago basado en protocolos de pago

En el caso de que el protocolo elegido requiera el intercambio de más de un mensaje, aparte de los mensajes intercambiados en el anterior apartado, haremos uso del método *MESSAGE* y del código de respuesta *200 (OK)* (ver Figura 5.3). Estos mensajes han sido extendidos para intercambiar la estructura *PaymentInformation* en su cuerpo. De esta manera, después del primer mensaje de pago enviado por el cliente en el mensaje *ACK*, el vendedor envía una respuesta en el método *MESSAGE*.

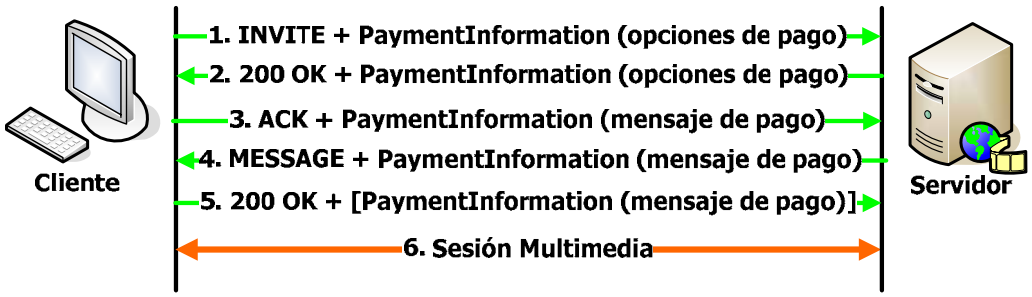


Figura 5.3. Pago con un protocolo de pagos.

Como respuesta a *MESSAGE*, el cliente envía el código de respuesta *200 (OK)* para confirmar la recepción del mensaje. Este mensaje, si fuera necesario, puede contener un mensaje de pago del cliente. Este par de mensajes *MESSAGE-200 (OK)* puede ser utilizado tantas veces como se necesite hasta completar el intercambio de mensajes del protocolo escogido.

El vendedor, con el último mensaje de pago, podría, opcionalmente, envíar información de fidelización o algún cupón que permita al cliente obtener algún beneficio (como un descuento) en el siguiente pago. El vendedor también puede enviar un justificante o recibo de la transacción. Una vez que el pago se ha llevado a cabo, la sesión comienza y el vendedor proporciona los contenidos/servicios pagados en la sesión. En el pago que acabamos de explicar, si el cliente hubiera solicitado un justificante de la transacción, éste lo recibiría en el mensaje que acabamos de describir.

Negociación del precio junto con las características de la sesión

En la anterior sección hemos supuesto que el precio estaba establecido por el vendedor y no podía ser negociado. Así, cuando el cliente recibe las opciones de pago con el precio en la descripción SDP, el cliente tiene dos opciones: o efectúa el pago o aborta la sesión. Esto es debido al hecho de que, en general, en las transacciones de comercio sobre servicios multimedia, el precio está establecido por el vendedor y no hay posibilidad de negociarlo. Sin embargo, hay vendedores que quieren soportar escenarios de negocio más ricos y características avanzadas como la negociación del precio junto con las características asociadas a la sesión (codecs, calidad, etc). De hecho, en el comercio electrónico B2C es un paso fundamental y valorado por los usuarios.

Podría suceder que el cliente considere que el precio solicitado por el vendedor no sea adecuado por el servicio solicitado. Así como también podría suceder que el vendedor pudiera ofrecer un descuento al cliente si éste posee un cupón o si es capaz de probar que pertenece a un grupo de usuarios que tiene reconocido un descuento específico. Por tanto, en estas situaciones, el cliente quiere negociar el precio con el vendedor.

La negociación como nueva característica en SIP

La negociación consiste en un intercambio de ofertas/contra-ofertas entre el cliente y el vendedor, y podría ser solicitado por el cliente. Por tanto, si queremos soportar esta característica en SIP, es necesario definir una nueva etiqueta de opción (option-tag) para especificar este comportamiento. Esta etiqueta de opción se ha denominado *payment.negotiation*. En concreto, esta opción se incluye en la solicitud *INVITE* que emite el cliente SIP. El soporte de la negociación no es obligatorio y el vendedor podría no estar dispuesto a aceptarla. En este caso, el vendedor no incluye, en el código de respuesta *200 (OK)*, la etiqueta *payment.negotiation* y también lo indica en el SDP extendido que contiene la información relacionada con el pago en la estructura *PaymentInformation*.

Proceso de negociación en SIP

El soporte de la negociación entre el cliente y el vendedor ha seguido la misma filosofía que la extensión de SIP para la fiabilidad de las respuestas provisionales (*Provisional Response Acknowledgement* – PRACK) [241]. Así, hemos decidido utilizar los tres

mensajes que componen esta extensión (*183 – Session in Progress*, *PRACK* y *200 (OK) a PRACK*) para incluir el SDP extendido descrito en la Sección 5.3.3. En estos mensajes, esta estructura contiene las opciones de pago soportadas así como el precio asociado. La indicación del deseo de negociación se incluye en la estructura *PaymentInformation*. Si la indicación no se incluye en la estructura, indicaría que es la última oferta del cliente.

El proceso de negociación consiste en una serie de intercambios de los mensajes *PRACK* y *200 (OK) a PRACK* (ver Figura 5.4) conteniendo una descripción SDP con la información de pago por la sesión completa o por cada uno de los flujos de datos. Cada uno de estos intercambios representa una oferta/contra-oferta. Además, en la estructura *PaymentInformation* ambas entidades pueden indicar si es la oferta/contra-oferta es su última propuesta por medio del atributo *negotiable* en los elementos *PaymentDescription*. Este mecanismo se utiliza para indicar que una de las partes no está dispuesta a seguir con la negociación. Así, el receptor debería aceptar la oferta o finalizar la negociación.

La negociación finaliza cuando o el cliente o el vendedor responden con un mensaje vacío o cuando uno de ellos decide abortar la sesión. Si el cliente acepta la última oferta del vendedor, debe de enviar un mensaje *PRACK* vacío. Por el contrario, si el vendedor acepta la última oferta del cliente, envía un código de respuesta *200 (OK) a PRACK* que no contiene nada en el cuerpo.

Cuando la negociación ha finalizado, el último paso del mecanismo básico de pago se ejecuta. Así, los mensajes *200 (OK)*, el *ACK* y *MESSAGE* (si es necesario) se usan para intercambiar la estructura *PaymentInformation* conteniendo los mensajes del protocolo de pago acordado en la negociación.

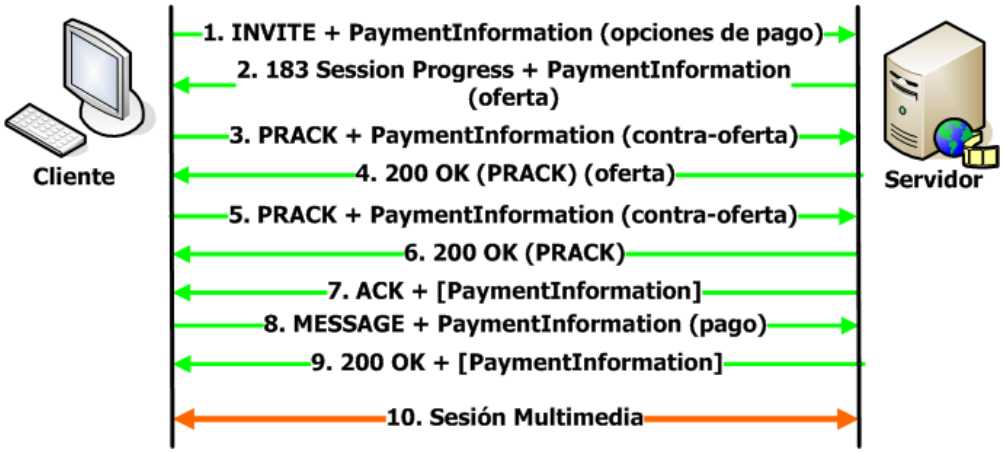


Figura 5.4. Negociación en SIP.

Recibos e información de fidelización

Después de realizar el pago por un servicio, algunas veces, el cliente requiere un recibo que justifique el pago realizado. La solicitud de este recibo puede ser enviada junto con el último mensaje de pago en el cuerpo de un mensaje con la estructura *PaymentInformation*.

Para soportar la emisión de recibos, también hacemos uso de la extensión que hemos propuesto para el método *MESSAGE*. Así, aparte de ser utilizado para enviar mensajes de pago, puede ser utilizado para contener recibos. Esta información de recibos se encapsula en la estructura *PaymentReceipts* de la estructura *PaymentInformation* (ver Apéndice F). De este modo, el vendedor envía un mensaje al cliente incluyendo el recibo después de recibir la solicitud del cliente. El vendedor también puede incluir, junto con el recibo, cierta información relacionada con la transacción como información de fidelización, un cupón o un ticket con un descuento para que el cliente obtenga determinados beneficios en las siguientes compras.

Pagos adicionales

Cuando se introdujo el mecanismo básico de pago para SIP, se suponía que el pago realizado por el cliente era por la sesión multimedia al completo. Dependiendo del tipo de sesión (calidad, duración, tipo de contenido multimedia, etc), la cantidad a pagar podría variar desde unos céntimos hasta una cantidad importante en Euros. Este enfoque es útil para aquellos vendedores que quieren obtener beneficios independientemente de si el usuario recibe todo el contenido de la sesión o no. Por ejemplo, el usuario, después de algún tiempo, se podría dar cuenta de que no le gusta el contenido/servicio o, por cualquier otra razón, podría tener que dejar la sesión.

Por otro lado, los clientes, normalmente, prefieren pagar por el contenido que obtienen. En las sesiones multimedia esto representa realizar pagos por una parte de la duración de la sesión en lugar de realizar un pago por toda la sesión independientemente del tiempo pasado. Finalmente, se podría seguir el enfoque *pay-as-you-watch* (*paga-lo-que-ves*) y que el pago se realizara conforme se va recibiendo el contenido. Así primero se recibe parte del contenido y, a continuación, se realiza el pago. El modelo se puede soportar si la implementación del servidor trata de controlar el bitrate con el que el contenido está siendo enviado al usuario, como se propone en [184].

Nuestra propuesta propone dividir el precio de la sesión en pequeñas unidades de tiempo/datos, etc. que puedan ser pagadas conforme el usuario desea acceder al contenido o una vez que ha accedido a él, dependerá del modelo. Así, después de cada pago, un intervalo de tiempo/datos está garantizado. En esta situación, el cliente tiene que ser avisado cuando el tiempo/datos por el que ha pagado está a punto de finalizar en el modelo *pay-per-view*, o se le tiene que requerir el pago por el contenido que ya ha recibido en el modelo *pay-as-you-watch*. En ambos casos, el pago se tiene que realizar al cabo de cada cierto tiempo y el usuario recibirá una alerta requiriendo el pago. Una vez que se recibe tal alerta, el cliente puede decidir si realiza otro pago o no. Está claro que no realizar otro pago supone el fin de la sesión. Este modelo en el que podemos realizar pagos durante la sesión es lo que hemos denominado el soporte de *pagos adicionales*.

El soporte de pagos adicionales en este framework se lleva a cabo por medio del método *MESSAGE*. En este caso, este mensaje contiene, en el cuerpo, una descripción SDP extendida donde podemos incluir información acerca del tiempo/datos que restan para finalizar la sesión. Para este propósito hemos definido un elemento llamado

StopInformation que puede ser incluido dentro del elemento *AdditionalInformation* de la estructura *PaymentInformation*.

5.3.3. Inclusión de información de pago en SDP

Para el soporte de este framework hemos comentado en la anterior sección, así como en la introducción de este capítulo, que es necesario poder especificar la información de pago junto con la descripción de la sesión multimedia. De esta forma pretendemos permitir la compra de productos multimedia que se distribuyen por medio de sesiones o bajo demanda. Los principales protocolos que utilizaremos para este fin son SIP y RTSP. Ambos tienen en común que para expresar la información asociada a la sesión multimedia utilizan el Protocolo para la Descripción de Sesiones (*Session Description Protocol* – SDP) [123]. Por tanto, para que ambos puedan manejar la información de pago asociada a la sesión, nuestro objetivo será ver cómo incorporar esta información de pago en SDP.

La actual especificación de SDP está diseñada para caracterizar las sesiones multimedia y los flujos de datos que la componen por medio de varios atributos. Sin embargo, hasta ahora, ninguno de estos atributos puede expresar información sobre pago.

Tal y como se recomienda en los sistemas de comercio electrónico y DRM es importante establecer la asociación entre la información de pago y los contenidos multimedia para facilitar el descubrimiento y los procesos de comparación. Esto también facilitaría el intercambio de esta información de una forma más eficiente ya que toda la información se enviaría de forma conjunta.

Para este propósito hemos decidido incluir la información de pago en las descripciones de los contenidos multimedia. Esta información de pago se utiliza para describir las opciones de pago así como, de forma opcional, las cantidades a pagar con cada opción.

La información de pago puede ser expresada a nivel de sesión o a nivel de flujo de datos dependiendo de si la sesión se paga como un todo o si es posible escoger los flujos de datos que componen la sesión. Además, hemos tenido en cuenta que cada flujo de datos podría ser transmitido con diferentes codecs y calidades. Por tanto, deberíamos poder especificar diferentes importes para los diferentes formatos multimedia.

Con el objetivo de incluir la información de pago en la forma que acabamos de describir, hemos extendido SDP con un nuevo atributo llamado *payment-info* que puede ser utilizado tanto a nivel de sesión como a nivel de flujo de datos. Su sintaxis es:

```
a=payment-info:codec:schema:PaymentInformation
```

o

```
a=payment-info:codec:uri:PaymentInformationURI
```

El elemento *codec* representa el número de codec indicado junto con el atributo *m* cuando lo estamos utilizando para asociar la información de pago a un flujo de datos. Si este atributo se utiliza a nivel de sesión, no hay número de codec.

El elemento siguiente a *codec* es una etiqueta para indicar si la información de pago está asociada junto con este atributo (etiqueta *schema*) o si la información de pago se referencia por medio de una URI (etiqueta *uri*). Esta URI podría apuntar a un documento externo en un servidor o podría apuntar a otro documento que se adjunta con este SDP en el cuerpo del mensaje del protocolo de aplicación que se utiliza para establecer la sesión multimedia. En general, se prefiere que la información de pago esté incluida junto con el atributo con el fin de evitar el establecimiento de una conexión adicional para obtener dicha descripción.

Independientemente del procedimiento seguido, al final el usuario obtiene una estructura *PaymentInformation* que describe la información pago que acabamos de mencionar. Esta estructura es un elemento XML cuya descripción completa se puede encontrar en el Apéndice F. Además, con este elemento, en SDP, podemos expresar las diferentes opciones de pago sin especificar la cantidad a pagar. De esta forma proporcionamos flexibilidad al vendedor para que, dependiendo del cliente, éste pueda ofrecer o negociar diferentes precios. Por ejemplo, el usuario podría pertenecer a un grupo específico de usuarios que tienen derecho a determinados descuentos (como los que mencionábamos en la Sección 3.2.7 cuando describíamos un escenario de compra de billetes de avión en un entorno de fidelización). Esta descripción SDP se intercambia con los protocolos de aplicación que veremos a continuación, es decir, SIP y RTSP.

5.3.4. Descripción detallada de las nuevas extensiones para SIP

En esta sección describimos, de forma detallada, las extensiones que hemos realizado para soportar el proceso de pago con SIP que se describió de forma genérica en la anterior sección. Del proceso explicado se desprende que para soportar el pago en SIP se han definido nuevas etiquetas de opción así como un nuevo tipo de contenido y se han extendido algunos de los mensajes que forman parte de SIP. Cada uno de estos elementos se explica a continuación.

Nuevas etiquetas de opciones

SIP permite que los mensajes puedan incluir distintas cabeceras para transportar la información que se necesita durante la sesión. Algunas cabeceras están compuestas por una lista de etiquetas de opciones que indican las características soportadas o requeridas. En concreto, la cabecera *Require* se utiliza para indicar las características que una de las partes tiene que soportar obligatoriamente, la cabecera *Supported* que se utiliza para indicar algunas características adicionales que se soportan y, finalmente, la cabecera *Accept* que indica los tipos de información que se soportan.

En este apartado, describimos el conjunto de nuevas etiquetas de opciones que hemos definido para soportar los distintos intercambios de información de pago de nuestro framework. Estas etiquetas son:

- *payment*. Se tiene que soportar obligatoriamente y es una nueva etiqueta para la cabecera *Supported*. Se ha definido para indicar que el cliente soporta el mecanismo básico de pago que proponemos, y que es lo mínimo que se necesita en el framework. Cuando se requiere un pago para acceder a una sesión multimedia, si el cliente no especifica esta etiqueta en el mensaje *INVITE*, el vendedor retornará un código de respuesta *402 (payment required)* con una descripción SDP que contiene la información de pago tal y como se describió en la Sección 5.2.
- *additionalpayments*. Es opcional. Esta nueva etiqueta se incluye en la cabecera *Supported* cuando el cliente quiere indicar que soporta realizar pagos adicionales en cualquier momento que se requiera a lo largo de la sesión. Con esta etiqueta el cliente también quiere indicar al vendedor que desea ser informado cuando el tiempo por el que pagó está llegando a su fin en el modelo *pay-per-view*. En el modelo *pay-as-you-watch* esto se interpreta como la solicitud del pago. Cuando el vendedor incluye esta etiqueta de opción, indica que podría solicitar pagos adicionales antes de que el periodo de tiempo pagado se acabe o para solicitar el pago una vez consumido parte del producto o servicio.
- *payment.negotiation*. Es opcional. Esta etiqueta ha sido definida para las cabeceras *Required* y *Supported*. Con ésta especificamos que el cliente soporta y/o quiere negociar el precio y la calidad de los flujos de datos de acuerdo al modelo oferta-respuesta explicado en la anterior sección. En concreto, en la cabecera *Required*, se indica que se quiere negociar. Por otro lado, en la cabecera *Supported* el cliente indica que soporta esta funcionalidad, aunque no la exige. Así se permite que el vendedor pueda iniciar una negociación y personalizar los precios en función del cliente.
- *application/payment*. El soporte de esta etiqueta es obligatorio para el framework de SIP. Esta etiqueta, que se define para la cabecera *Accept*, indica a la otra parte que puede intercambiar información de pago de acuerdo a la extensión que hemos definido. Por tanto, al incluirla en esta cabecera indicamos que hemos definido un tipo de contenido que será insertado en el cuerpo de un mensaje con la cabecera *content-type* marcada como *application/payment*.

Tipo de contenido application/payment

Hemos definido un nuevo tipo de contenido para el cuerpo de los mensajes SIP. Este contenido se referencia indicando que su *content-type* es *application/payment*, tal y como hemos mencionado en la anterior sección. En concreto, se utilizará para recibir las distintas opciones de pago y/o enviar los mensajes de pago con la estructura *PaymentInformation*.

Extensiones a los mensajes SIP

En esta sección especificamos en detalle cómo hemos utilizado las extensiones de SIP en los distintos mensajes. También explicaremos cómo hemos incluido la información de pago en cada uno de los mensajes para llevar a cabo los procesos descritos en la sección

anterior. Los distintos mensajes que pueden ser intercambiados en un flujo de ejecución aparecen en la Figura 5.5. Comentaremos cada uno de los mensajes conforme aparecen en la figura, pero nos centraremos en explicar cómo incluir la información de pago ya que el proceso fue explicado en la anterior sección.

Opciones de pago soportadas

Para el intercambio de las opciones de pago soportadas en el mensaje *OPTIONS* (mensaje X. Todos los mensajes que se mencionan en este apartado harán referencia a la Figura 5.5), en la cabecera *Accept*, debe aparecer la etiqueta de opción *application/payment*. Opcionalmente, el cliente puede enviar las opciones de pago que soporta en el cuerpo del mensaje con la estructura *PaymentInformation* indicando en la cabecera *content-type* el valor *application/payment*. En este caso los elementos involucrados en la estructura son: *PaymentProtocols*, *Brands*, *PaymentServiceProviders* and *PaymentDescriptions*. En este caso las descripciones contenidas en *PaymentDescriptions* no especifican ningún precio.

Como respuesta a esta consulta, el vendedor emite un mensaje (mensaje Y) proporcionando la información de pago solicitada. Esta información se incluye de la misma forma que en el mensaje anterior.

Inicio de sesión con negociación

Para iniciar una sesión que da acceso a unos contenidos que requieren un pago, el cliente envía un mensaje *INVITE* (mensaje 1). En este mensaje, se debe de incluir, en la cabecera *Supported*, la etiqueta *payment* para iniciar que el cliente soporta el mecanismo de pagos básico.

Opcionalmente, el cliente puede incluir, en la cabecera *Supported*, la etiqueta *additionalpayment* para indicar que también soporta el mecanismo de pagos adicionales. Además, el cliente puede especificar la etiqueta *payment.negotiation* para indicar que quiere negociar el precio de las opciones de pago de los flujos de datos de la sesión. La otra posibilidad para la etiqueta *payment.negotiation* es incorporarla en la cabecera *Supported*. En este caso, la semántica indica que el cliente soporta y podría estar interesado en la negociación. Opcionalmente, el cliente puede enviar las opciones de pago con las que trabaja junto con la descripción SDP extendida. La información de pago de los métodos soportados se obtendrá de los wallets que el cliente tiene y que su especificación es conforme a la proporcionada en el capítulo anterior.

Como respuesta, el vendedor emite el código de respuesta *183 (Session Progress)* (mensaje 2) que contiene en el cuerpo la descripción SDP extendida los precios de las diferentes opciones de pago. En este caso el *content-type* se marca como *application/sdp*, tal y como se define en el estándar de SIP. Esta descripción contiene, para la sesión completa o por cada flujo de datos, el atributo *payment-info* que incluye la solicitud de pago en la estructura *PaymentInfornation*.

El cliente, para realizar una contra-oferta, emite un mensaje *PRACK* (mensaje 3). La oferta se envía en el cuerpo como una descripción SDP extendida. El vendedor puede enviar una contra-oferta con el código de respuesta *200 (OK) a PRACK* y en el cuerpo del

mensaje la descripción SDP extendida. En caso de que no se envíe contra-oferta el cuerpo está vacío.

Para dar por finalizada la negociación se envía el código de respuesta *200 (OK)* (mensaje 5). Este mensaje contiene la descripción SDP con la información final sobre el pago a realizar por la sesión entera o por cada stream. Esta descripción se envía en el cuerpo del mensaje de forma similar a los mensajes anteriormente comentados.

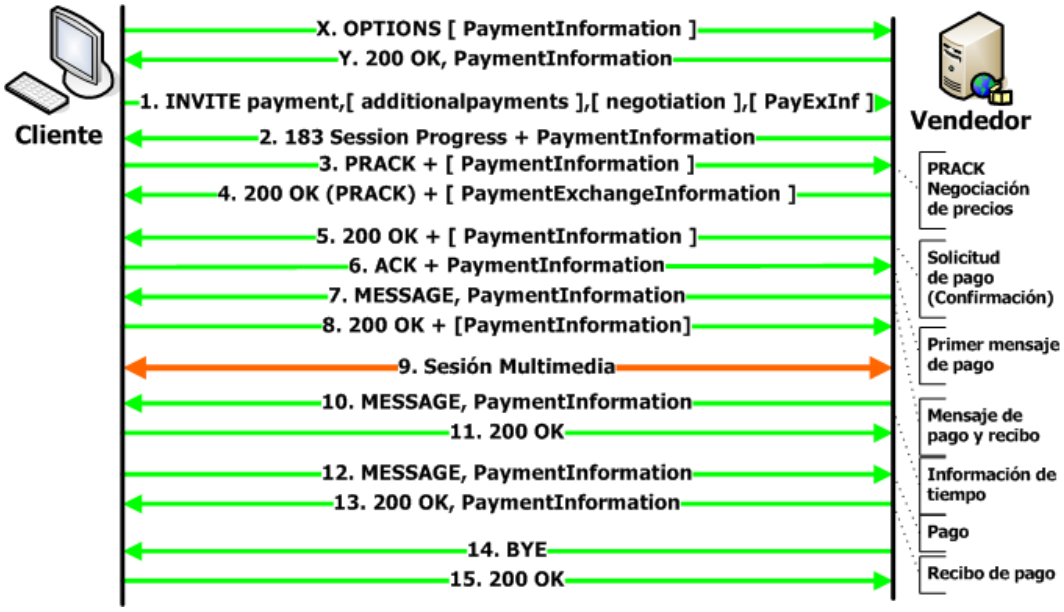


Figura 5.5. Flujo de mensajes en el framework de pagos con SIP.

Intercambio de información de pago

Los mensajes 6, 7 y 8 han sido extendidos para permitir el intercambio de información de pago (mensajes de pago, credenciales, información de fidelización, etc) que se necesita para acceder a la sesión y finalizar el proceso de inicialización. A continuación, hemos indicado cómo se ha extendido para mensaje para incluir dicha información.

El mensaje *ACK* (mensaje 6) la información se adjunta en el cuerpo de este mensaje con la estructura *PaymentInformation* indicando en la cabecera *content-type* el valor *application/sippayment*. En esta estructura, el cliente utiliza el campo *PaymentProtocolMessage* para enviar un mensaje de pago, o el campo *Receipts* para enviar el recibo de haber efectuado el pago a otra entidad. Estos campos se rellenan a partir de la información generada por el wallet del mecanismo de pago concreto. Los mensajes 7 (MESSAGE) y 8 (código de respuesta *200 OK)* se pueden utilizar tantas veces como sea necesario en el caso de que se requiera intercambiar información adicional antes de producirse el envío del flujo de información. La información se incluye de la misma forma que en el mensaje *ACK*. También en el mensaje 8 se podría enviar un justificante de pago.

Envío de información multimedia

Una vez que se ha llevado a cabo el pago, se inicia el intercambio de los diferentes flujos (mensaje 9) entre cliente y vendedor. En este caso no se muestra ningún intercambio en el flujo de mensajes porque depende del protocolo y los medios acordados.

Pagos adicionales

Para el soporte del mecanismo denominado pagos adicionales, se intercambian los mensajes comprendidos entre el 10 y 13.

En este caso, cuando el tiempo por el que pagó está a punto de terminar o cuando se requiere el pago por lo reproducido, el vendedor le envía un mensaje del tipo *MESSAGE* indicando este hecho (mensaje 10). Esta información se envía dentro de la estructura *TimeToStop* que está contenida en el elemento *AdditionalInformation* de la estructura *PaymentInformation*. Esta estructura se envía en el cuerpo del mensaje con la descripción SDP extendida que contiene los precios a ser pagados. Como respuesta, el cliente envía el código de respuesta *200 (OK)* (mensaje 11), que no necesita incluir ninguna información adicional.

Si el cliente, finalmente, decide continuar con la sesión y realizar un nuevo pago, utiliza el método *MESSAGE* (mensaje 12) y el código de respuesta *200 (OK)* (mensaje 13) del mismo modo descrito para los mensajes 7 y 8.

Fin de sesión

Para la finalización de la sesión no ha sido necesario realizar ninguna extensión. Así, el mensaje *BYE* y su respuesta (mensajes 14 y 15) se intercambian para finalizar la sesión conforme se especifica en SIP.

5.3.5. Casos de uso

En esta sección describimos como el framework podría ser utilizado para acceder algunos servicios multimedia de pago por medio de dos protocolos de pago. En primer lugar, mostraremos el uso de un protocolo de moneda electrónica como PURSE-COIN. En segundo lugar, utilizaremos un protocolo de macropagos como SET. Para cada escenario describimos los distintos mensajes a intercambiar. Por simplicidad, en estos escenarios suponemos que no hay negociación del precio.

Pago con moneda electrónica

Supongamos que una compañía de VoIP (Voz sobre IP) proporciona un servicio de mensajes de voz donde la gente puede escuchar los mensajes de voz que tiene en su buzón. Los términos en los cuáles se ofrece este servicio los describimos a continuación. En primer lugar, hay un pago inicial para acceder al servicio durante dos minutos. Después de que la sesión se haya iniciado, se requiere un nuevo pago por cada minuto adicional.

En este ejemplo supongamos que el cliente sólo necesita un minuto adicional para escuchar sus mensajes. Los mensajes que forman parte de este ejemplo aparecen en la

Figura 5.6. No explicaremos en detalle ni los mensajes ni las extensiones porque ya fueron explicados en las secciones anteriores.

El cliente inicia la sesión con el mensaje *INVITE* e indicando que soporta Payword [238], PURSE-COIN [251, 252], Paycash [218, 219], D-Cash [148], cheques electrónicos de viaje [171] y SET [191, 274] (paso 1). Como respuesta, el vendedor envía el código de respuesta *200 (OK)* (paso 2) con la descripción SDP que contiene la información de pago que acabamos de mencionar. En este caso, el vendedor sólo envía los precios para Payword, PURSE-COIN y D-Cash que son los que éste soporta. Supongamos que el precio más económico es con PURSE-COIN y Payword. Junto con estas opciones se envía determinada información que serviría para la inicialización del pago en PURSE-COIN. Esta información es el identificador del vendedor (*VendorID*), el identificador de la transacción (*TransID*) y la fecha (*Date*). Como el cliente los soporta todos, elige entre los que ofrecen el precio más económico, en este caso Payword y PURSE-COIN. Como PURSE-COIN ofrece mejor nivel de seguridad que Payword se decanta por PURSE-COIN.

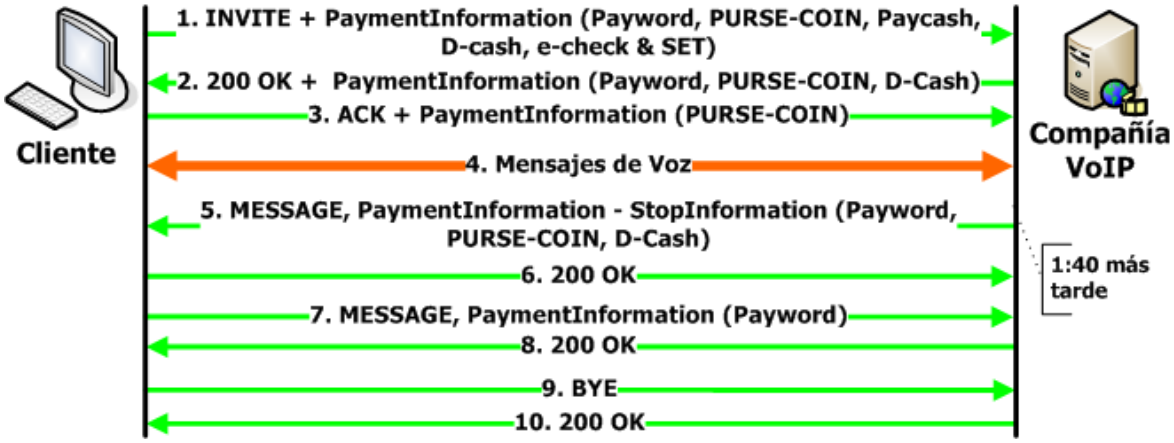


Figura 5.6. Pago en SIP con un protocolo de moneda electrónica.

A continuación, se invoca al wallet (conforme a la especificación dada en el Capítulo 3) del protocolo PURSE-COIN para que genere una moneda a partir de la información de inicialización recibida. El usuario envía esta moneda en el mensaje *ACK* (paso 3). A partir de ese momento el usuario comienza a escuchar los mensajes (paso 4). Este pago permite al usuario escuchar los mensajes durante un minuto. En este ejemplo también hemos supuesto que el cliente no ha solicitado un recibo.

Cuando el tiempo por el que pagó el usuario está a punto de finalizar (por ejemplo, cuando apenas quedan veinte segundos), el vendedor envía una solicitud de pago con el método *MESSAGE* que contiene una descripción SDP indicando los precios que indicamos para este minuto adicional (paso 5). Como respuesta a este aviso, el cliente confirma que lo ha recibido (paso 6). En este caso, como hemos comentado anteriormente, suponemos que el cliente desea continuar y realizar el pago. Por tanto, envía otra moneda generada

con PURSE-COIN (paso 7). El servidor, en el siguiente mensaje, confirma la recepción del mensaje (paso 8).

Finalmente, cuando el usuario ha finalizado la reproducción de sus mensajes de voz, la sesión finaliza (mensajes 9 y 10).

Pago con SET

Supongamos que una compañía quiere ofrecer el acceso a una conferencia virtual con dos presentaciones. Para el inicio de la sesión y poder participar en la primera conferencia el vendedor requiere un pago inicial. Una vez que el tiempo del pago inicial se ha agotado, a continuación, el vendedor solicitará un nuevo pago que dará derecho a recibir la segunda conferencia.

En este escenario, suponemos que el cliente soporta los mismos protocolos que en caso anterior. Sin embargo, el vendedor, debido a que suponemos que la cantidad a recibir corresponde al importe de un macropago, sólo soporta protocolos de macropagos, como SET o cheques de viajes. En este escenario, finalmente suponemos que el cliente elige el protocolo SET. En este protocolo, el intercambio entre el cliente y el vendedor, como vimos en el Capítulo 2, está compuesto de cuatro mensajes: *Initiate Request*, *Initiate Response*, *Purchase Request* y *Purchase Response*. Entre el vendedor y la pasarela de pago, además, se intercambiarían los mensajes *Authorization Request/Response* y *Capture Request/Response*.

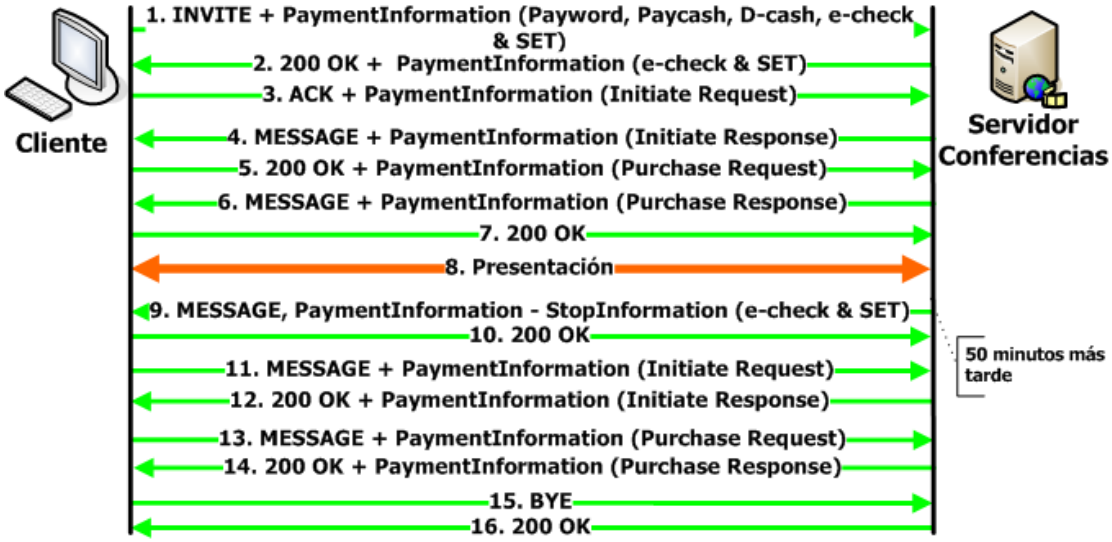


Figura 5.7. Pago con SET.

La sesión se inicia con el intercambio de las opciones soportadas (mensajes 1 y 2, ver Figura 5.7. En este apartado todos los mensajes hacen referencia a esa figura.). Ahora, el vendedor sólo ofrece SET y los cheques de viaje. El cliente decide utilizar SET y en el mensaje *ACK* envía el primer mensaje de este protocolo (*Initiate Request*) (mensaje 3). El resto de los mensajes del protocolo se intercambian por medio del método *MESSAGE*

(mensajes 6 y 7) como podemos ver en la Figura 5.7. En la figura, por simplificar, no se han incluido los mensajes del servidor de conferencias con la pasarela de pagos para los mensajes de SET.

Una vez que se ha producido el pago, el usuario accede a la primera presentación. Más tarde, cuando la sesión esté a punto de finalizar (por ejemplo, cuando sólo queden diez minutos), el vendedor envía un aviso al cliente y le solicita un nuevo pago. El usuario, en este caso, decide pagar por la segunda presentación y, por tanto, el protocolo de pago se ejecuta (mensajes del 11 al 14). Finalmente, después de la segunda presentación, la sesión finaliza (mensajes 15 y 16).

5.3.6. Comparación con la propuesta de Jenning et al.

En esta sección comparamos nuestra propuesta con la propuesta más similar existente en el trabajo previo y que es un draft del IETF propuesto por Jenning et al. [146]. Su objetivo, al igual que nosotros, es soportar el pago bajo SIP.

Como mencionamos en la Sección 5.2.3, dicha propuesta no es adecuada para micropagos porque incluye la participación de una tercera parte que realiza la transferencia entre el banco del cliente y el banco del vendedor con altos costes de transacción. Tampoco soporta el uso de diferentes protocolos de pagos, ya que como acabamos de mencionar, está basada en la transferencia de fondos entre el banco del cliente y el del vendedor. Además, la entidad que recibe el pago (el PSP) conoce las relaciones entre los clientes y los vendedores y no permite que los clientes utilicen protocolos de pago anónimos.

Con el fin de satisfacer los objetivos planteados en la introducción de este capítulo podríamos haber tomado esta propuesta como base e intentado extenderla con los mismos mecanismos y estructuras planteados en las Secciones 5.3.2 y 5.3.4. Sin embargo, tal y como se nuestra en la comparación que vamos a realizar a continuación, nuestra propuesta es más eficiente en cuanto al número de mensajes a intercambiar (la información de pago intercambiada y los mensajes de pago suponemos que son los mismos).

En esta comparación, para la que utilizaremos los protocolos PURSE-COIN y SET, en primer lugar comparamos sólo él mecanismo de pago básico, donde sólo hay un pago al principio de la sesión. Posteriormente, realizaremos la comparación teniendo en cuenta que se produce el pago inicial y un pago adicional.

En la comparación no hemos tenido en cuenta los mensajes para acceder a los contenidos/servicios multimedia una vez que el pago se ha realizado. Tampoco se han tenido en cuenta los mensajes con la pasarela de pagos SET porque en ambos casos el número de mensajes sería el mismo.

La comparación está basada en los escenarios que hemos explicado en la anterior sección ya que representan tanto el uso de un protocolo de micropagos como uno de macropagos.

Tal y como se muestra en la Tabla 5.1, en el modo básico, la propuesta de Jennings et al. necesita cuatro mensajes más para el caso de PURSE-COIN y seis más para SET. En

el modo con pago adicional, en PURSE-COIN necesitaría en ambos caso cuatro mensajes más.

También es importante mencionar que nuestra propuesta puede utilizar el uso de los recibos generados por la tercera parte de Jennings et al. y podrían ser enviados en la nuestra para realizar un pago.

		PURSE-COIN	SET
Pago básico	Jennings et al.	9	15
	Nuestra propuesta	5	9
Con Pago Adicional	Jennings et al.	13	19
	Nuestra propuesta	9	15

Tabla 5.1. Comparación basada en el número de mensajes.

5.3.7. Seguridad

En cuanto a la seguridad del proceso de pago, comentar que el framework proporciona la misma seguridad que el protocolo utilizado para realizar la compra del servicio multimedia. Así, si el método soportado (por ejemplo, una moneda electrónica eCash) ofrece seguridad contra falsificación o doble gasto, el framework también las proporciona porque la seguridad reside en el método de pago subyacente.

Como se analizó en el Capítulo 2, los sistemas de micropagos, en general, no garantizan el intercambio equitativo porque en estos sistemas la pérdida es pequeña y el aspecto más importante es la eficiencia. Los vendedores que ofrecen estos esquemas prefieren proporcionar los flujos de datos como un conjunto de partes con un valor muy bajo que es pagado individualmente en lugar de pagar el flujo de datos al completo. Cuando un nuevo pago se efectúa, una nueva parte del flujo se recibe. En estos escenarios de micropagos, los vendedores, habitualmente, están interesados en tener tantos clientes como sea posible y no engañarlos por una cantidad pequeña. El framework sigue el mismo enfoque. Si el método de pago no es un micropago, el protocolo de pagos elegido debería de garantizar, de alguna forma, el intercambio equitativo y esta información se podría insertar en los mensajes definidos en el framework.

Adicionalmente, con el fin de intercambiar de forma segura la información de negociación y/o pago entre las partes se pueden utilizar cualquiera de los mecanismos comentados en [221] como son IPSEC y TLS (a nivel de red o para securizar el transporte a nivel TCP), S/MIME o MIKEY (a nivel de mensaje SIP) y SIPS (para URIs seguras). Por otro lado, para el transporte seguro de la información en tiempo real se puede utilizar SRTP o TLS. Finalmente, los mensajes de mensajería instantánea se podría securizar mediante TLS o por medio de S/MIME.

5.3.8. Conclusiones

Gracias a la mejora en las infraestructuras de comunicaciones y a las velocidades ofrecidas por las distintas tecnologías de red hoy se hace posible el plantearnos la distribución de contenidos multimedia que se distribuyen bajo sesiones.

El soporte de pagos en SIP es interesante para realizar la compra de estos nuevos contenidos y servicios que progresivamente van apareciendo y que se suministran de este modo. A lo largo de la Sección 5.3, hemos presentado cómo SIP puede ser extendido, basado en los mecanismos de extensibilidad que ofrece, para soportar el pago de sesiones multimedia de una forma genérica. Para este fin hemos hecho uso de la extensión de SDP propuesta en la Sección 5.3.3 y hemos extendido SIP para incluir la información de pago.

Nuestra propuesta satisface los objetivos establecidos al principio de este capítulo y puede trabajar tanto con protocolos de micropagos como de macropagos, así como cualquier otro sistema de pago, incluso propietarios. Es más, soporta el intercambio de información definido por otras propuestas como, por ejemplo, la de Jennings et al. El soporte de estos pagos se realizaría por medio de los wallets definidos en el capítulo anterior. La extensión propuesta es especialmente adecuada para protocolos basados en micropagos ya que no requiere de intercambios adicionales a los que por defecto se necesitan en SIP.

Adicionalmente, nuestra propuesta ofrece otras características interesantes como la negociación tanto de la calidad de los flujos de datos como los precios asociados y el soporte de pagos adicionales durante la sesión sin tener que reiniciar la sesión. En la negociación hemos tenido en cuenta que el precio de la transacción puede variar dependiendo de las opciones de pago elegidas.

Para soportar esas características, hemos soportado y extendido el modelo oferta-respuesta, la descripción de SDP y algunos métodos de SIP. La sobrecarga introducida por nuestra extensión es mínima y la participación de terceras partes no es necesaria aunque se soporta. El objetivo de nuestras extensiones a SIP ha sido mantener el protocolo tan simple como sea posible.

Por tanto podemos concluir que ofrecemos una solución de pago en SIP que facilita su incorporación a las actuales implementaciones. Además, permite ofrecer distintas soluciones de pago que pueden ser utilizadas de otros entornos como los desarrollados en el framework de pago presentado en el Capítulo 4. A su vez, éstas podrán reutilizarse en futuros escenarios que puedan aparecer.

5.4. Framework de pagos para RTSP

5.4.1. Introducción

En esta sección explicaremos cómo hemos añadido el soporte de pagos para el protocolo Real Time Streaming Protocol (RTSP) [270]. El hecho de proponer un framework de pagos para RTSP persigue satisfacer los objetivos planteados al principio de este capítulo para el caso de los contenidos que se distribuyen bajo demanda. La distribución para este

tipo de contenidos se realiza por medio de un protocolo ampliamente extendido como es RTSP.

Así en esta sección veremos cómo hemos proporcionado soporte para efectuar pagos por servicios multimedia descargados bajo demanda en Internet. Para esta solución nos hemos basado en las mismas ideas que aportamos para el framework de SIP. Así, pretendemos integrar los pagos en RTSP para que los desarrolladores de RTSP puedan ofrecer soluciones interoperables y en las que la integración sea independiente del mecanismo de pago. Además, para facilitar su aceptación e incorporación a las soluciones existentes nuestra propuesta está basada en los mecanismos de extensibilidad de RTSP Así se permite a los vendedores ofrecer una amplia variedad de protocolos de pago y modelos de negocio. Además, se pretende que los wallets y la información de pago se utilicen conforme fueron definidos en el capítulo anterior. Así los wallets y los mecanismos de pago podrían ser utilizados en distintos escenarios sin tener que volver a desarrollarlos para cada escenario en particular.

Al igual que en SIP, a la hora de especificar la información de pago, también buscaremos facilitar la especificación de diferentes precios para los flujos de datos con diferente calidad. Es más, para un flujo de una determinada calidad debería ser posible especificar un precio diferente por cada posible opción de pago.

Finalmente, buscaremos proporcionar para este escenario, al igual que en SIP, la realización de pagos adicionales de forma que se pueda efectuar un pago mientras el contenido está siendo enviado como flujos bajo demanda (streaming). De esta manera, permitiremos realizar un pago inicial por, digamos una parte inicial del contenido y, posteriormente, si el cliente le gusta el contenido, él haría un pago adicional para continuar recibiendo el flujo de datos sin ninguna interrupción.

Debido a las similitudes con la propuesta para SIP, no explicaremos los mecanismos descritos ya anteriormente ni mostraremos ningún caso de uso ya que las opciones a mostrar serían muy similares a las mostradas en la Sección 5.3.5, con las diferencias debidas al protocolo a utilizar y el contenido a distribuir. En las siguientes secciones sólo entraremos en detalle en los aspectos diferenciadores, el resto de los detalles se pueden consultar en [259, 261].

5.4.2. Visión genérica del framework de pagos RTSP

El framework que proponemos a continuación se basa en la extensión que se ha propuesto de SDP (ver Sección 5.3.3), y en la extensión de RTSP que introducimos en esta sección y de la que proporcionaremos más detalles en las siguientes secciones. Esta extensión está basada en las mismas ideas que fueron presentadas para el framework de pagos en SIP en la anterior sección. De hecho, esta propuesta pretende satisfacer los mismos requisitos que se establecieron en la Sección 5.3.1, pero en lugar de para el establecimiento de sesiones multimedia, para la distribución de contenidos bajo streaming.

Para la extensión de RTSP nos hemos basado en los mecanismos de extensibilidad que éste soporta [270]. Este hecho permite la compatibilidad con soluciones desarrolladas por otros vendedores así como que las actuales implementaciones de RTSP puedan incorporar

la solución aquí propuesta y, por tanto, permite que más usuarios sean clientes potenciales. Así, nuestro framework facilita el desarrollo de un middleware que diferentes clientes de RTSP podrían usar para efectuar pagos. El framework además permite a los clientes pagar por solamente lo que reproduzcan.

En este framework hemos definido dos modos de operación de forma similar a cómo se hacía también en SIP: el modo básico y el modo avanzado. En el modo básico, el usuario efectúa un pago por el flujo de datos completo. Cuando el flujo de datos llega a su fin, la sesión finaliza. En el modo avanzado, el usuario lleva a cabo un pago inicial antes o después de recibir el contenido (según si el modelo es pay-per-view o pay-as-you-watch) y, posteriormente, puede realizar pagos adicionales para reproducir el contenido sin interrupciones. De esta forma, el cliente puede pagar sólo por el contenido que realmente está interesado.

Al igual que en SIP, este framework también ofrece otras posibilidades como consultar la información de la sesión junto con las opciones de pago antes de iniciar la sesión RTSP (en este protocolo, a diferencia de SIP, define mensajes para la consulta de la información de sesión antes de su inicio), la negociación del precio de las opciones de pago y la calidad de la sesión y el envío de recibos e información de fidelización.

5.4.3. Extensión de RTSP

En esta sección describiremos, de forma detallada, las fases que componen nuestro framework de distribución de contenidos bajo demanda. En primer lugar, explicaremos cómo es posible consultar la información que describe la sesión y las opciones de pago asociadas, a continuación veremos cómo realizar la negociación del precio y de la calidad, después cómo realizar la inicialización de la sesión a la misma vez que realizamos el pago básico, después veremos cómo incorporar la información de recibos e información de fidelización y finalmente, cómo realizar los pagos adicionales. En esta sección además realizaremos también algunas consideraciones acerca de la seguridad del sistema.

Por cada fase, proporcionaremos los detalles acerca de las extensiones realizadas o utilizadas en RTSP para soportar nuestro framework e incluir información de pago. La Figura 5.8 muestra los distintos mensajes que se intercambian en las diferentes fases que pueden tener en el framework. Estas fases se comentan en detalle a lo largo de los siguientes apartados.

Solicitud de la descripción de la información de pago del contenido multimedia

De forma previa al pago de la sesión, el cliente podría querer conocer alguna información relacionada con el proceso de pago. En primer lugar, las opciones de pago soportadas por el vendedor y, en segundo lugar, el precio de la sesión (o de sus flujos de datos) con cada una de las opciones de pago. Toda esta información podría ser recuperada o bien de un servidor Web (con HTTP/HTTPS) o bien del servidor RTSP donde la información multimedia está almacenada (con el método *DESCRIBE* como explicaremos después).

Independientemente del método utilizado para recuperar la información, al final, el cliente obtiene una descripción SDP extendida, tal y como se comentó en la Sección 5.3.3.

Soporte de pagos y opciones de pago soportadas

El soporte de pagos podría ser visto como una característica o capacidad adicional del servidor RTSP. En RTSP, el modo para descubrir las capacidades de un servidor, un cliente o un proxy es por medio del método *OPTIONS* (mensajes W y X de la Figura 5.8. Todos los mensajes en esta sección se referirán a esta figura).

Para tratar el pago como una capacidad del servidor hemos definido tres nuevas etiquetas de opciones o características (*option-tags*) para la cabecera *Supported* del método *OPTIONS*. Con estas etiquetas indicaremos el soporte de pagos tanto en solicitudes como en respuestas así como el soporte de la negociación de la calidad y el precio. Estas etiquetas son: *play.payment, play.additionalpayments* y *payment.negotiation*. También hemos definido una nueva etiqueta para la cabecera *Accept* y que hemos denominado *application/payment*. Esta etiqueta se utilizará para indicar que en un determinado mensaje el contenido del cuerpo contendrá información de pago de la misma forma que definíamos para SIP.

La etiqueta *play.payment* indica que el cliente, el vendedor o el proxy soportan pagos en el método *PLAY*. Este método es el mínimo que tiene que implementar el servidor si quiere recibir pagos conforme a este framework. Para el soporte de este método también se requiere el soporte de la etiqueta *application/payment*. Las otras dos características se consideran opcionales.

El cliente incluye *play.additionalpayments* para indicar que después del pago inicial, podría realizar pagos adicionales, y por tanto, quiere que el vendedor le informe antes de que el tiempo por el que pagó finalice. El vendedor utiliza esta etiqueta para indicar que soportan este mecanismo.

La etiqueta *payment.negotiation* se utiliza para negociar el precio y la calidad con la que se emitirá un determinado contenido. Esta negociación se iniciará con el método *DESCRIBE*.

Finalmente, la etiqueta *application/payment* del método *Accept* indica que se permitirá el intercambio de información de pago en el cuerpo de los mensajes de forma similar a como explicábamos para SIP.

Si el servidor sólo ofrece contenidos en el modo pay-per-view, al menos la etiqueta *play.payment* deberá aparecer en la cabecera *Require*, en lugar de aparecer en la cabecera *Supported*.

Adicionalmente, el cliente puede preguntar sobre las opciones de pago soportadas por el servidor RTSP. Para este fin, también extendemos el método *OPTIONS* añadiendo al cuerpo del mensaje un nuevo contenido, cuyo *content-type* es *application/payment* (ver mensajes Y y Z). Así el cuerpo del mensaje contiene el elemento *PaymentInformation*. En este caso en el campo *PaymentDescriptions* aparecerían las combinaciones válidas de las distintas opciones de pago soportadas. Además, en la cabecera *Accept* se indicaría la etiqueta *application/payment*. En la respuesta del vendedor, el cuerpo del mensaje *200 (OK)* incluiría un contenido del tipo *application/payment* conteniendo la estructura

PaymentInformation con las opciones de pago soportadas. A este cuerpo con el *content-type application/payment* y con la estructura *PaymentInformation*, de aquí en adelante, para simplificar, lo denominaremos cuerpo de pago.

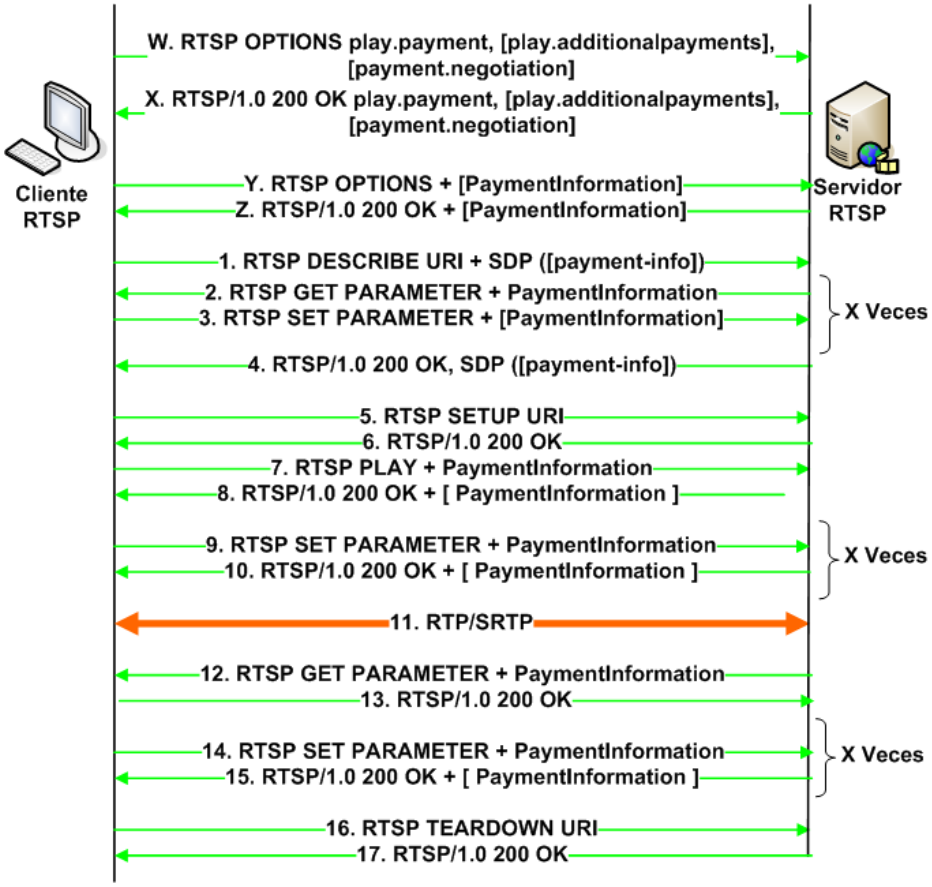


Figura 5.8. Framework de pagos RTSP.

Descripción de la sesión multimedia

El cliente puede solicitar la descripción de la sesión con el método *DESCRIBE* (mensaje 1). La respuesta contiene una descripción SDP que incluye la información de pago (mensaje 4).

Adicionalmente, en la solicitud (mensaje 1) el cliente puede enviar las opciones de pago que soporta y el orden en que prefiere utilizarlas. La lista de opciones de pago se envía en la solicitud por medio del SDP extendido que permite indicar las opciones de pago a nivel de sesión o a nivel de stream. Junto con esta lista, el cliente podría incluir información adicional ligada a una opción de pago concreta, como parámetros de inicialización.

Si el cliente incluye la lista en la solicitud, el vendedor sólo ofrecerá en la descripción SDP las opciones de pago indicadas por el cliente, o un subconjunto de ellas. Si el vendedor no soporta ninguna de ellas, enviará una respuesta con un error *501 (Not*

implemented). La información de pago que proporciona el vendedor se incluye en la descripción SDP con el atributo *payment-info* (ver Sección 5.3.3).

Negociación del precio y de la calidad de la sesión

En el anterior intercambio de mensajes hemos supuesto que o el cliente no estaba interesado en negociar el precio del contenido multimedia o el vendedor no permitía esta opción. Sin embargo, podría ser que una de las partes si estuviera en la negociación del precio, principalmente el cliente.

En el caso de que se quisiera realizar la negociación, el cliente cuando envía la solicitud para obtener la descripción multimedia, aparte de enviar la descripción SDP extendida tendría que indicar que quiere (en la cabecera *Require*) o soporta (en la cabecera *Supported*) la negociación mediante la etiqueta *payment.negotiation*.

Si el cliente exige negociar y el cliente no está dispuesto a negociar, como respuesta enviaría un código de error *503 (Service unavailable)*. En caso contrario, la negociación se iniciaría con el mensaje *GET_PARAMETER (mensaje 2)*. El vendedor, en el cuerpo de este mensaje, enviaría una descripción SDP extendida indicando para cada uno de los codecs y para cada una de las opciones de pago, el precio asociado. El cliente enviaría la contra-oferta de forma similar con el mensaje *SET_PARAMETER* (mensaje 3). Estos dos mensajes los podrían utilizar el cliente y el vendedor tantas veces como sea necesario hasta que lleguen a un acuerdo o alguno de ellos decida terminar la negociación.

En cualquier momento de la negociación, cualquiera de las entidades puede indicar que es su última oferta y que no está dispuesta a seguir con la negociación. Esta opción se indica en la estructura *PaymentInformation* junto con las opciones de pago con el atributo *negotiable*.

La aceptación de los términos de la negociación por parte del cliente se indica mediante el envío de un mensaje *SET_PARAMETER* con el cuerpo del mensaje vacío. A continuación, el vendedor envía un mensaje *200 (OK)* (mensaje 4) para finalizar el método *DESCRIBE* y conteniendo el acuerdo alcanzado con la descripción SDP extendida. Este mensaje es el que también envía el vendedor en el caso de que sea él quién decide aceptar la oferta del cliente.

Inicialización de la sesión: pago y reproducción

A partir de la descripción SDP extendida, el cliente puede elegir los flujos de datos y las opciones a pagar. Es importante señalar que a diferencia del modelo oferta-respuesta donde todos los flujos de datos deberían ser reproducidos, aquí, permitimos la compra de unos determinados streams que forman parte de la sesión multimedia.

Como define el estándar RTSP, el cliente establece la sesión mediante el envío del método *SETUP* (mensajes 5 y 6). Después, en el flujo habitual de RTSP, el cliente solicita la reproducción del contenido con el método *PLAY*. En nuestra propuesta hemos decidido mantener este enfoque y extender este último método para enviar la información de pago.

El cliente inicia el proceso de pago con el método *PLAY* (mensaje 11) o para la sesión entera o para un flujo de datos particular. Hemos decidido realizarlo en el método *PLAY*

en lugar de en el método *SETUP* porque éste es el momento en el que el usuario realmente pide la reproducción del contenido. Es más, es justo el momento cuando la reproducción tiene lugar. Otra de las opciones considerada fue la introducción de un nuevo método en RTSP para realizar el pago, pero rechazamos esta idea ya que los actuales métodos de RTSP ofrecen suficientes mecanismos de extensibilidad para soportar su mejora. Así, la incorporación de estas nuevas características a las actuales implementaciones de RTSP es más sencilla y no requiere mensajes adicionales.

Como acabamos de comentar, el método *PLAY* inicia el proceso de pago que nos dará acceso al contenido en el modelo pay-per-view. En el modelo pay-as-you-watch, el proceso sería similar salvo que ahora el pago se realizaría después de haber recibido cierto contenido. Para intercambiar los mensajes del protocolo de pago generados a partir de su wallet correspondiente utilizamos el cuerpo de pago.

Cuando el vendedor recibe esta estructura (mensaje 7), comprueba si la información es correcta o no. Si es correcta envía un mensaje con el método *200 (OK)* (mensaje 12). En caso contrario, envía el código de error *401 (Payment required)*. En el método *200 (OK)*, el servidor también incluye, en el cuerpo de pago, el mensaje de respuesta del protocolo de pago (mensaje 8).

Normalmente, en los sistemas de micropagos el proceso de pago finaliza después de estos dos intercambios (mensajes 7 y 8). Por tanto, a continuación el vendedor envía los flujos de datos al cliente (mensaje 15). Por otro lado, para algunos protocolos de pago se requiere el intercambio de más mensajes entre el cliente y el servidor. Para enviar estos mensajes de pago adicional proponemos el uso del método *SET_PARAMETER* (mensaje 9) y su respuesta asociada en el *200 (OK)* (mensaje 10) con el cuerpo de pago. Estos mensajes pueden ser enviados tantas veces como se necesite hasta que el protocolo de pago finalice. Después de enviar el último mensaje *200 (OK)*, la reproducción del contenido comienza (mensaje 11).

El vendedor envía los flujos de datos al cliente durante el tiempo pagado. Si el cliente y el vendedor sólo soportan la característica *play.payment*, la sesión finaliza después de este tiempo. Por otro lado, si tanto el cliente y el vendedor soportan la característica *play.additionalpayment* (ya sea para pagos adicionales como para el modelo *pay-as-you-watch*), el proceso podría continuar como se describe posteriormente.

Recibos e información de fidelización

Aparte de los mensajes de pago, este framework también soporta el intercambio de otra información relacionada con el pago, como los recibos de compra o la información de fidelización.

Si el cliente quiere un recibo de la transacción, éste puede solicitarlo a la misma vez que envía el último mensaje de pago (mensaje 7) por medio del elemento *Receipts* que se encuentra en *PaymentInformation*. El vendedor incluirá el recibo en el mensaje que envía como respuesta al mensaje de pago (mensaje 8).

La información de fidelización que el usuario posee se envía al principio del intercambio de los mensajes de pago (mensaje 5). En este caso, el cliente envía, por medio de la

estructura *PaymentInformation* la información de fidelización en el elemento *LoyaltyInformations* así como el mensaje de pago. El vendedor puede proporcionar nueva información de fidelización con cualquier mensaje de pago. Para este propósito el vendedor también usa la estructura *LoyaltyInformations*.

Pagos adicionales

Cuando se soporta la característica *play.additionalpayment*, el servidor avisa al cliente cuando el tiempo por el que pagó está cerca de finalizar (modelo *pay-per-view*) o cuando se requiere que se efectúe un pago por el tiempo/datos consumidos (modelo *pay-as-you-watch*). El tiempo de anticipación con el que el servidor informará no está preestablecido y podría variar dependiendo del tráfico en la red, el tiempo que podría llevar realizar el pago o basado en las preferencias del usuario.

El servidor solicita un nuevo pago utilizando el método *GET_PARAMETER* (mensajes 12 y 13). En el elemento *PaymentInformation*, dentro del campo *AdditionalInformation*, el vendedor enviará una estructura denominada *TimeToStop*. Este elemento contiene información acerca del tiempo restante y, opcionalmente, se podría enviar nueva información acerca de las opciones para llevar a cabo el pago. Si no hay nueva información acerca de las opciones de pago, se considera que son las mismas que se utilizaron para realizar el pago inicial. La estructura *PaymentInformation* se envía en el cuerpo del mensaje como en anteriores mensajes.

En el modelo *pay-per-view*, una vez que el cliente recibe la alerta sobre la inminente finalización del tiempo, éste podría ignorar la advertencia. En este caso, cuando el tiempo acaba, el servidor para el envío del streaming. En caso contrario, si al cliente le gusta el contenido multimedia, éste podría realizar un pago adicional. Sin embargo, el cliente no debe enviar otra vez el mensaje *PLAY* porque esto supondría la interrupción de la reproducción actual. Para evitar este problema, enviamos la información de pago en el método *SET_PARAMETER* (mensaje 14). Este es el mismo intercambio que tuvo lugar en los mensajes 9 y 10.

En el modelo *pay-as-you-watch*, el cliente tendría que realizar el pago tal y como se acaba de explicar. A continuación, si no desea seguir recibiendo contenidos pasaría a finalizar la sesión como se explica en el siguiente apartado. En caso contrario, el cliente continúa recibiendo contenidos y posteriormente, se volvería a solicitar un pago. Si el vendedor no recibe el pago la sesión finaliza.

Fin de la sesión

Cuando el tiempo por el que pagó el usuario acaba, o el contenido se ha reproducido por completo, el servidor para el envío de flujos de datos. Después, el cliente cierra la sesión con el método estándar *TEARDOWN* (mensajes 20 y 21), donde no ha sido necesario incluir ninguna extensión.

5.4.4. Seguridad

De forma similar a otras propuestas que hemos presentado como EPP y el Framework de pagos de SIP. Esta propuesta es independiente del protocolo de pago, por lo que la seguridad del sistema dependerá del esquema elegido.

De forma complementaria, las extensiones de RTP/RTSP, tales como MIKEY y SRTP, podrían ser usadas con este framework para proporcionar la privacidad de la información enviada bajo demanda. Además, también se pueden aplicar las extensiones S/MIME a los mensajes RTSP para garantizar la autenticidad y/o confidencialidad de la información intercambiada con el protocolo (las descripciones SDP conteniendo la información de pago, la información de pago de un determinado protocolo, negociación, etc).

Otro aspecto interesante es la autenticación del flujo de datos. Si es necesario, podríamos seguir un enfoque similar al propuesto en [297]. Este aspecto no está relacionado con el nivel de pago y, por tanto, no afecta a los mensajes propuestos en el framework.

5.4.5. Conclusiones

En esta sección hemos presentado un nuevo framework de pagos para servicios bajo demanda en Internet basados en RTSP. Para su soporte ha sido necesario utilizar la extensión planteada para SDP así como extender determinados mensajes de RTSP e incluir nuevos atributos.

En el framework, la información de pago está ligada con las solicitudes de contenido multimedia. Este framework fue diseñado especialmente para el soporte de los micropagos, ya que para estos sistemas no requiere del intercambio de mensajes adicionales a los que por defecto se necesita en RTSP, aunque soporta cualquier otro sistema de pagos. Además, permite seguir el modelo oferta-respuesta con el fin de soportar la elección y negociación de la calidad de los streams y las opciones de pago, pudiendo indicar por cada opción de pago una cantidad diferente. Finalmente, como última característica a mencionar es la posibilidad de negociar el precio a pagar.

Por tanto, proporcionamos un framework basado en el estándar RTSP que permite incorporar, en escenarios de distribución de contenidos bajo demanda, el uso de distintos sistemas de pago de una forma genérica. También hemos visto cómo los mecanismos ofrecidos pueden soportar distintos modelos de negocio.

El pago se realiza por medio del wallet del protocolo correspondiente. Este wallet sigue la especificación presentada en el Capítulo 4. De esta forma, los distintos desarrolladores de clientes y servidores de distribución bajo demanda pueden soportar el pago de una forma interoperable. Además, el trabajo a los desarrolladores de sistemas de pago se simplifica ya que la provisión de sistema de pagos sería mediante el wallet propuesto en el Capítulo 4. Así, el mismo sistema de pago se puede utilizar en distintos escenarios y frameworks.

5.5. Conclusiones del capítulo

A día de hoy, cada vez más, los usuarios están interesados en productos y servicios electrónicos multimedia. En particular, nos estamos refiriendo a productos electrónicos que se distribuyen bajo demanda y a productos o servicios que se suministran una vez que se ha establecido una sesión SIP. Dentro de estas categorías podemos encontrar productos como son noticias, vídeos musicales o películas bajo demanda, juegos on-line, VoIP, etc. Los proveedores de estos contenidos o servicios se pueden plantear el querer obtener beneficios de la provisión de éstos. Como hemos presentado a lo largo de este capítulo, el proceso de pago para estos productos o servicios ha tenido una cobertura limitada. Por este motivo encontramos soluciones que están basadas en mecanismos de pago particulares o soluciones que no son lo suficientemente genéricas para soportar distintos mecanismos de pago.

Como respuesta a estas necesidades y a partir de los resultados del capítulo anterior, en este capítulo hemos presentado una solución diferente para cada entorno. Esta diversidad está motivada por el hecho de que para los contenidos distribuidos bajo demanda y para los contenidos o servicios basados en SIP existen soluciones estandarizadas que permiten su acceso y control. En concreto, RTSP y SIP, respectivamente. Por tanto, decidimos que el mejor enfoque era extender, siempre que fuera posible, estas soluciones para evitar tener que realizar conexiones adicionales entre las distintas partes que incrementaran los costes computacionales. De hecho, aunque hubiéramos decidido utilizar otro enfoque como un framework de pagos genérico, como el presentado en el capítulo anterior para tales productos, finalmente, hubiera sido necesario extender dichos protocolos para enviar algún tipo de información referente al pago. Además, este enfoque tampoco sería tan eficiente como si aprovecháramos los mecanismos de extensibilidad de los protocolos diseñados para tales productos.

En la Figura 5.9 se presentamos las aportaciones que hemos realizado en este capítulo. Estas aportaciones, como se puede observar complementan a las soluciones presentadas anteriormente y que se mostraban en la Figura 4.18. En concreto, en este capítulo, las aportaciones aparecen en los dos niveles superiores. En el primer lugar hemos incorporado la extensión a SDP. A continuación, en el siguiente nivel, están las extensiones realizadas sobre SIP y RTSP.

El hecho de que las soluciones propuestas estén basadas en los mecanismos de extensibilidad de los distintos protocolos existentes también facilita su incorporación a las soluciones hasta ahora desarrolladas para dichos protocolos. Así, las distintas soluciones y paquetes software que soportan estos protocolos pueden incorporar las propuestas aquí presentadas de una forma más sencilla que si nuestras propuestas implicarán importantes cambios en cuanto al funcionamiento de esos protocolos.

De las soluciones presentadas podemos destacar varias características como que permiten utilizar distintos protocolos de pago, por cada una de éstos pueden expresar diferentes precios, permiten la negociación de las características del producto así como del precio, pueden soportar distintos modelos de negocio y, finalmente, para el envío de los

mensajes de pago se utilizan los mensajes de los protocolos necesarios para el control del acceso al producto o servicio.

También cabe destacar que las soluciones están basadas en los componentes que fueron introducidos en el capítulo anterior. De hecho, para describir la información de pago utilizamos los elementos XML que se definieron, para el uso de distintos protocolos se pueden utilizar los wallets y, finalmente, para el intercambio de los mensajes de pago se utilizarán los wallets conforme se presentó en el Capítulo 4. Por tanto, se prueba que los componentes anteriormente presentados son lo suficientemente genéricos para que puedan ser utilizados en escenarios de pagos como los presentados en este capítulo así como otros posibles futuros escenarios que puedan aparecer. De esta forma, se facilita la reusabilidad de los componentes de software de pago y su introducción en distintos escenarios ya que el mismo wallet podría ser utilizado para el pago de un fichero PDF, un vídeo musical, el acceso a un juego en línea, etc.

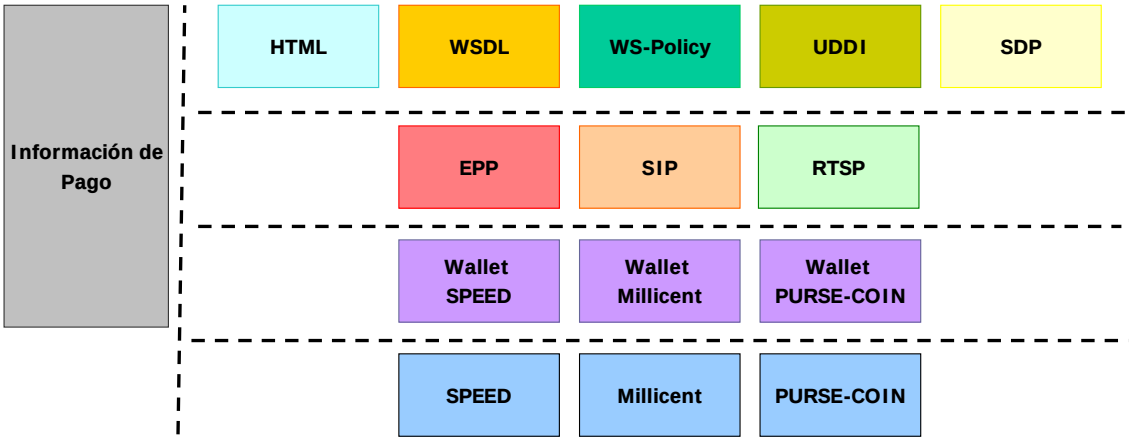


Figura 5.9. Visión genérica de las soluciones aportadas en esta tesis.

Capítulo 6

Conclusiones y vías futuras

Este último capítulo de la tesis doctoral tiene dos objetivos fundamentales: por un lado, presentar las principales conclusiones que se pueden extraer a partir de las diferentes propuestas introducidas en este trabajo de investigación; por otro lado, pretende señalar cuáles son las posibles líneas de investigación futuras que se derivan del trabajo aquí presentado y que permitirán mejorar o ampliar los beneficios de los resultados aquí obtenidos.

6.1. Conclusiones

Como se ha mencionado a lo largo de esta tesis algunos de los principales factores por los que el comercio electrónico no ha tenido el éxito esperado son: la falta de confianza del usuario en los sistemas de pago, sistemas complejos de manejar, interfaces no comunes y/o amigables, seguridad de los sistemas, falta de pilotos que muestren al usuario el funcionamiento de los sistemas de comercio, metodologías para analizar y modelar la seguridad de las transacciones de negocio.

En respuesta a esta problemática, en esta tesis nos hemos centrado en dos aspectos fundamentales relacionados con la adquisición de productos o servicios electrónicos en entornos de comercio B2C. Por un lado, en el desarrollo de sistemas de pago seguros que permitan generar confianza en el usuario; por otro lado, en el desarrollo de interfaces comunes y de frameworks que faciliten el uso de los sistemas de pago, permitiendo ofrecer un entorno uniforme para la realización de pagos que genere confianza en el usuario.

En concreto, en esta tesis en primer lugar hemos realizado un análisis de las distintas propuestas que han ido apareciendo a lo largo del período en el que se ha realizado esta investigación. De este análisis, la principal conclusión que hemos obtenido es que a pesar de ser un tema profundamente estudiado y en el que se han presentado un amplio número de propuestas, existen determinados aspectos por cubrir en cuanto a los sistemas de pagos (principalmente relacionados con la seguridad) y los frameworks asociados a éstos (definición de todos los componentes necesarios, protocolo genérico con negociación, etc). De este análisis también cabe mencionar que ofrece un amplio estado del arte acerca de las distintas propuestas para estos campos.

A continuación, teniendo en cuenta la problemática asociada, hemos abordado el problema de las soluciones de pago en el Capítulo 3. Posteriormente, en los Capítulos 4 y 5, hemos trabajado en frameworks de pagos para distintos tipos de productos y servicios.

Con respecto a los sistemas de pago, en esta tesis hemos presentado dos soluciones de pago: SPEED y PURSE-COIN. Ambas soluciones se basan en el uso de los monederos electrónicos ya que éstos pueden aportar características interesantes como mayor seguridad que las tarjetas de banda magnética, facilitan la movilidad del dinero electrónico y pueden ser utilizados en diversos escenarios de pago: máquinas de vending, pagos por Internet, etc. El hecho de ofrecer dos propuestas de pago basadas en monedero electrónico se debe a que queremos facilitar el uso desde distintos puntos de vista. La primera propuesta está enfocada a ofrecer un sistema de pago basado en monedero en Internet que permita la adquisición de contenidos electrónicos garantizando el no repudio, el intercambio equitativo y la atomicidad. Estos contenidos electrónicos se envían de forma cifrada de forma que sólo el cliente y el vendedor pueden acceder al contenido. La segunda propuesta pretende incrementar la aceptación de pagos basados en monedero eliminando la necesidad del módulo SAM para la recepción de pagos electrónicos.

Por tanto podemos concluir que estas propuestas:

- Ofrecen protocolos que facilitan el uso de los monederos electrónicos para la realización de pagos en la Web.
- Tanto SPEED como PURSE-COIN combinado con SPEED proporcionan soluciones de pago para la adquisición de contenidos electrónicos que contemplan la seguridad en las principales fases del proceso de compra: negociación, pago y entrega del producto.
- Facilitan el soporte de pagos basados en monedero electrónico a los vendedores ya que en el caso de SPEED el pago lo recibe una tercera parte que es la encargada de integrar los dispositivos con los módulos SAM. En el caso de PURSE-COIN no es necesario el uso de tales dispositivos hardware.
- Proporcionan un mayor nivel de seguridad que las propuestas presentadas hasta la fecha garantizando el no repudio, el intercambio equitativo y la atomicidad.
- La seguridad de estas propuestas ha sido validada formalmente por medio de una herramienta de validación automática como AVISPA que permite analizar distintos ataques de seguridad conforme al modelo Dolev-Yao.
- SPEED ha sido desarrollado y evaluado en distintos escenarios que han permitido probar las características mencionadas.

A partir de estas soluciones se ha pasado a abordar el problema de ofrecer frameworks de pago que permitan la negociación y elección de los mecanismos de pago a utilizar para realizar la adquisición de un determinado producto o servicio electrónico.

Inicialmente el problema se abordó para los productos que se podían adquirir a través de la Web con un solo clic. En el diseño de esta solución se definieron varios componentes para facilitar el intercambio de información de pago, su negociación y, finalmente, poder

efectuar el pago. Entre estos componentes destacan el protocolo de pago extendido denominado EPP que soporta la negociación de las opciones de pago a utilizar y una vez que éstas han sido elegidas se utiliza para enviar los mensajes del protocolo de pago elegido. Este protocolo está orientado a sesión y permite soportar varios modelos de negocio. Asociado a este protocolo se definió un wallet genérico que permite el uso de distintos protocolos a partir de una interfaz común ofreciendo así un mecanismo de abstracción que permite a EPP utilizar distintos protocolos de pago de una manera uniforme sin tener que preocuparse por las particularidades de un determinado protocolo en particular. Finalmente, se definió, de forma genérica, la información necesaria para realizar el intercambio de las opciones de pago. Esta información nos ha permitido a su vez definir los enlaces de pago. Estos enlaces ayudarán a identificar los productos de pago y a realizar el descubrimiento y comparación de características y precios de los productos y servicios.

Estas ideas se aplicaron al acceso de servicios Web basados en pago. Para facilitar el descubrimiento y el procesamiento de la información de pago asociado a éstos fue necesario definir cómo anotarlos. Una vez descubiertos los servicios, el proceso de pago se podía llevar a cabo por medio protocolo genérico de pagos EPP.

Finalmente, las ideas del framework de pagos por clic a continuación se aplicaron a la adquisición de otros productos/servicios electrónicos que contaban con un determinado protocolo de control tales como los servicios multimedia basados en SIP o los contenidos multimedia distribuidos por streaming con RTSP. De esta forma a la misma vez que se realiza el control de la sesión multimedia se puede intercambiar la información de pago.

Las principales conclusiones que se derivan de los frameworks que hemos definido se mencionan a continuación:

- Los frameworks de pago genéricos aportados definen todos los elementos necesarios para llevar a cabo la elección del protocolo de pago y el posterior pago.
- Se ha definido un protocolo genérico de pagos que está orientado a sesión, permite la negociación de las opciones a utilizar y el precio asociado, soporta distintos modelos de negocio y encapsula los mensajes del protocolo de pagos elegido. Este protocolo ha sido implementado, evaluado y la sobrecarga introducida se ha mostrado aceptable.
- El diseño de wallet definido se puede utilizar en distintos escenarios y permiten que los frameworks se abstraigan de las particularidades de cualquier protocolo de pago. Además, su utilización se ha probado con dos protocolos de pago: SPEED y Millicent.
- Los frameworks ofrecen un modo uniforme de uso de los protocolos de pago y del envío de los mensajes asociados a éstos.
- La información de pago ha sido definida de forma genérica con el fin de que se pueda utilizar en distintos frameworks a la misma vez que facilita su descubrimiento y procesamiento automático.
- Los frameworks permiten la adquisición de los principales tipos de productos/servicios electrónicos que se pueden contemplar en la red: contenidos

electrónicos que se acceden a través de la Web (páginas Web, documentos electrónicos, ficheros de música, etc), servicios Web, servicios multimedia que se establecen por medio de SIP (VoIP, juegos electrónicos, servicios basados en mensajería instantánea, etc) y, finalmente, contenidos que se distribuyen bajo streaming por medio de RTSP.

- Las soluciones ofrecidas en los distintos frameworks están basadas en los mecanismos de extensibilidad de los distintos estándares que serán utilizados en un entorno particular lo que facilita su desarrollo y aceptación.

Como conclusión global al trabajo realizado en esta tesis doctoral podemos afirmar que hemos proporcionado una serie de soluciones que están encaminadas a facilitar el uso de los sistemas de pago electrónico para llevar a cabo la adquisición de productos y servicios electrónicos en los entornos de comercio electrónico B2C. Estas soluciones cubren las necesidades que existen dentro del ámbito de los sistemas y frameworks de pagos que, aunque hemos podido comprobar que han sido estudiados durante un largo período de tiempo, todavía presentan aspectos a mejorar.

Los protocolos de pago presentados, que mejoran las propuestas de pago ya existentes, están basados en el uso de monederos electrónicos que aportan mayor seguridad y conveniencia al usuario a la hora de utilizar los sistemas de pago electrónico. Las soluciones al estar basadas en monederos electrónicos facilitan su uso en diferentes escenarios como el pago en máquinas de vending o POS así como para pagos en remoto a través de Internet. Es importante destacar este último escenario ya que es donde las propuestas anteriores presentaban mayores problemas.

Finalmente, destacar que esta tesis ofrece una solución que contempla el proceso de pago de una forma global, de forma que no se centra en un protocolo de pagos concreto, sino que además se proporcionan los mecanismos necesarios para el uso de distintos protocolos (bien los existentes o cualquier otro que pueda aparecer en el futuro aportando nuevas características de seguridad o modelando nuevos requisitos de negocio) mediante los distintos frameworks definidos. Por tanto, nuestros frameworks son extensibles y están abiertos a nuevas posibilidades. Los frameworks definidos, además, a diferencia de los trabajos que se han realizado previamente definen todos los componentes necesarios para facilitar la interoperabilidad entre las distintas soluciones de pago y permitir la elección del protocolo de pago a utilizar.

6.2. Vías futuras

En este trabajo de tesis doctoral hemos presentado una serie de soluciones para facilitar el uso protocolos de pago y mejorar la seguridad de éstos. Sin embargo, las soluciones aportadas no presentan la solución definitiva a estos problemas. A lo largo de este trabajo de investigación nos hemos podido dar cuenta de que, a partir de éste, se plantean nuevas mejoras, retos y líneas de investigación. A continuación pasaremos a describir brevemente cuáles son estas posibilidades. En primer lugar, comentaremos las mejoras que se pueden

incorporar a las propuestas presentadas y, a continuación, las líneas de investigación que se derivan de este trabajo.

6.2.1. Mejoras y ampliaciones

Con respecto a los protocolos de pago aquí presentados, hemos visto como nuestras propuestas garantizan el no repudio mediante el uso de firmas electrónicas. Estas firmas suponen una importante carga computacional en los dispositivos como los monederos electrónicos. Una de las posibilidades a investigar es nuevas primitivas o algoritmos de firma que permitan realizar este proceso de firma electrónica de forma más segura y eficiente (tanto en tiempo como en el tamaño de la firma genera) como podría ser la incorporación de criptografía de curva elíptica [197, 278].

Otro escenario que tendría que ser más estudiado sería el uso de PURSE-COIN combinado con las tarjetas SIM para ofrecer soluciones de pago móvil [126, 127]. Decíamos que la solución propuesta en PURSE-COIN es una applet Java Card que podría ser instalado en una tarjeta SIM. Para ofrecer una solución de pago móvil habría que analizar si, a partir de soluciones como SATSA [284] es posible crear una interfaz genérica de pago que permita la realización realizar de transacciones de pago desde nuestro teléfono móvil de una forma sencilla y uniforme. De esta forma se podrían desarrollar aplicaciones para móviles para realizar pagos por medio de SMS [289]. Además, al desarrollar aplicaciones de este tipo podríamos incrementar la seguridad del sistema y poder establecer un canal confiable entre la tarjeta inteligente y el usuario, mejorando la autenticación del cliente y eliminando el POS de la cadena de confianza [143].

Además, habría que estudiar las distintas posibilidades a la hora de realizar la transferencia de este dinero desde un teléfono móvil a otro teléfono móvil (pagos persona a persona) o a una máquina de vending de forma que se soportaran distintos tipos de escenarios [319, 320]. De esta forma además estaríamos ofreciendo soluciones para los entornos de comercio móvil. El trasladar las soluciones definidas a estos entornos constituye un reto debido a las restricciones de los sistemas móviles, sobre todo de los teléfonos móviles, en cuanto a capacidades criptográficas, velocidades de transmisión, etc. Aunque también es cierto que cada vez estas restricciones suponen un problema menor debido a los importantes avances que se están produciendo en el hardware de estos dispositivos, de forma que cada vez disponemos de dispositivos con mayores capacidades en menor tamaño.

Otro aspecto a tratar dentro de este apartado sería el desarrollo de una ontología que modele los principales conceptos relacionados con la información de pago y que permita que los elementos XML puedan ser anotados con los conceptos de esta ontología. De esta forma, se facilitaría el descubrimiento de la información en la Web.

6.2.2. Líneas futuras

Hasta ahora la mayoría de las soluciones de pago estaban centradas en el modelo de pago tradicional en el que el pago se lleva a cabo de un cliente a un vendedor (directa o indirectamente). Sin embargo, a día de hoy, han surgido escenarios más complejos en el

que un cliente necesita realizar un pago con múltiples proveedores de servicios al mismo tiempo (lo que se conocen como protocolos de pago multiparte) [41, 42]. En estos escenarios será interesante estudiar cómo llevar a cabo estos pagos por medio de las soluciones basadas en monederos electrónico a la misma vez que se tienen en cuenta los distintos aspectos relacionados con el no repudio y el intercambio equitativo [210]. Otro aspecto que también hay que considerar en los sistemas de pago y que cada vez tiene más importancia es la privacidad [233].

En nuestros sistemas de pago uno de los pilares es el uso de tarjetas inteligentes. Recientemente han aparecido otros dispositivos con características similares denominados Módulos de Plataforma Confiable (TPM – Trusted Platform Module) [110]. Estos dispositivos son similares a las tarjetas inteligentes ya que son un módulo de seguridad de bajo coste implementado como un dispositivo resistente a ataques de manipulación. Estos módulos han sido específicamente diseñados para ser la base de la computación confiable (trusted computing) [93, 187, 264]. Los TPMs están ligados a una plataforma específica a diferencia de las tarjetas que se pueden utilizar a través de múltiples sistemas. Una línea interesante será estudiar en detalle cómo esta nueva tecnología se puede aplicar para la realización de pagos [14, 15] y ver en qué medida las ideas en las que se basa PURSE-COIN pueden ser aplicadas en este entorno. Incluso se puede estudiar en determinados entornos cómo estos dos tipos de dispositivos podrían ser utilizados de forma complementaria para proporcionar diferentes tipos de soluciones de pago.

Relacionado con estos escenarios, incluso se podría estudiar si es posible extender PURSE-COIN de forma que sea posible realizar pagos cliente a cliente, de forma que se pueda transferir dinero de un cliente a otro sin necesidad de establecer contacto con el emisor. De esta forma, estudiaríamos en más detalle los distintos requisitos, condiciones y ventajas de la realización de pagos persona a persona [319, 320]. En el anterior apartado ya se introducía este tema desde el punto de vista del uso PURSE-COIN y las tecnologías de transferencia de red, aquí este punto se abordará desde un punto de vista más genérico que llevaría a su extensión o a la realización de una nueva propuesta.

Desde el punto de vista de los frameworks de pagos se abren distintas líneas de investigación a seguir. En los frameworks que hemos presentado, comentamos que uno de los principales objetivos es que éstos fueran independientes del protocolo finalmente elegido para realizar el pago. Por tanto, la seguridad del sistema dependía principalmente del protocolo de pagos utilizado para llevar a cabo el pago. Sin embargo, es interesante estudiar si independientemente del nivel de seguridad ofrecido, es posible que el framework pueda ofrecer un nivel de seguridad adicional. El objetivo será analizar si, en determinadas circunstancias, el framework proporcione evidencias adicionales al protocolo de pagos utilizado y que éstas puedan probar que la transacción tuvo lugar o al menos que se garantice que determinada información se intercambió. Por tanto sería necesario estudiar las última técnicas de no repudio para su aplicación a la generación de protocolos y frameworks de pagos [129]. Este escenario además es especialmente interesante para el framework de pagos basado en contenidos que se distribuyen bajo streaming y para el framework de pago basados en sesiones multimedia [88, 134, 137]. En ambos casos el producto o servicio a proporcionar no está claramente delimitado ni se pueden generar

evidencias del todo ya que el contenido a transmitir no es un fichero al completo sino que son una serie de flujos cuya duración y contenido podría no estar completamente determinado. En este caso habría que estudiar el modelo pay-as-you-watch en más detalle para ver cómo combinarlo para generar las evidencias que permitan demostrar, en aquellos casos en que sea necesario, que se envío (o no) determinada información.

Otro framework de pagos que podría considerar como caso de estudio sería aplicar las ideas presentadas anteriormente en un escenario donde no hay un solo vendedor sino que hay múltiples clientes y múltiples vendedores como podría ser en un escenario de distribución de contenidos P2P [175]. Asociado a este escenario habría que analizar, si es posible, cómo y qué tipo de protocolos utilizar para llevar a cabo el pago por productos que se adquieren de esta forma. Relacionado con este escenario y de forma complementaria al pago habría que tener en cuenta los pagos derivados de los derechos de autor asociadas a cada contenido que se distribuye mediante este sistema. El pago por los derechos de autor también constituye un reto en los otros frameworks presentados.

Finalmente, con respecto al framework de pagos basado en SIP, éste se debería analizar en más detalle con respecto a su uso dentro del framework para la entrega de servicios multimedia en redes IP (IMS – IP Multimedia System) ya que éste forma parte de las especificaciones de los dispositivos conforme a las especificaciones 3GPP/3GPP2/TISPAN. El objetivo sería analizar las posibilidades de uso dentro de este sistema con el fin de ofrecer soluciones de pago genéricas que se puedan utilizar en dispositivos móviles [125, 186]. De esta forma, se podría extender el uso de sistemas de pago móviles. Hay que tener en cuenta que los dispositivos móviles están ampliamente extendidos y en algunos países el índice de penetración es superior al 100%. Por tanto, la posibilidad de incorporar el uso de sistemas de pago en dispositivos móviles por medio de mecanismos uniformes y ampliamente extendidos proporcionará ventajas tanto a clientes, vendedores como a bancos y pondrán disponer de un sistema de pago móvil que siempre estaría disponible a los usuarios y que se podría utilizar en cualquier lugar.

Bibliografía

[1] M. Abadi, y R. Needham, “Prudent Engineering Practice for Cryptographic Protocols”, IEEE Trans. Softw. Eng., vol. 22, no. 1, pp. 6-15. 1996.

[2] D. Abrazhevich, “Classification and Characteristics of Electronic Payment Systems”, Proceedings of the Second International Conference on Electronic Commerce and Web Technologies, Springer-Verlag, pp. 81-90. 2001

[3] N. Adachi et al., “The Security Problems of Rivest and Shamir's PayWord Scheme”, Proceedings of the International Conference on Commerce and Enterprise Computing 2003, pp. 20. 2003.

[4] G. Agnew, “Secure electronic transactions: overview, capabilities, and current status”, Payment technologies for E-commerce, Springer-Verlag New York, Inc., pp. 211-226. 2003.

[5] A. Andrieux et al., ”Web Services Agreement Specification (WS-Agreement)”. https://forge.gridforum.org/projects/graap-wg/. Accedido: 19 de Diciembre de 2.008.

[6] APACS, “Fraud. The Facts 2007”. 2007. http://www.apacs.org.uk/resources_publications/documents/FraudtheFacts2007.pdf. Accedido: 9 de Marzo de 2009.

[7] A. Armando y L. Compagna, “SATMC: A SAT-Based Model Checker for Security Protocols”, Logics in Artificial Intelligence, pp. 730-733. 2004.

[8] A. Armando, L. Compagna, y P. Ganty, “SAT-Based Model-Checking of Security Protocols Using Planning Graph Analysis”, FME 2003: Formal Methods, pp. 875-893. 2003.

[9] A. Armando et al., “The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications”, Computer Aided Verification, pp. 281-285. 2005.

[10] N. Asokan et al., “The State of the Art in Electronic Payment Systems”, Computer, vol. 30, no. 9, pp. 28-35. 1997.

[11] Avispa Project: “Automated Validation of Internet Security Protocols and Applications (AVISPA)”. http://www.avispa-project.org/. Accedido: 15 de Septiembre de 2.008.

[12] S. Bajaj et al., “Web Services Policy 1.2 - Attachment (WS-PolicyAttachment)”. W3C. http://www.w3.org/Submission/WS-PolicyAttachment/. Accedido: 19 de Diciembre de 2.008.

[13] S. Bajaj et al., “Web Services Policy 1.2 - Framework (WS-Policy)”. W3C. http://www.w3.org/Submission/WS-Policy/. Accedido: 19 de Diciembre de 2.008.

[14] S. Balfe, y K.G. Paterson, “e-EMV: emulating EMV for internet payments with trusted computing technologies”, Proceedings of the 3rd ACM workshop on Scalable trusted computing 2008, pp. 81-92. 2008.

[15] S. Balfe, y K. Paterson, “Augmenting Internet-Based Card Not Present Transactions with Trusted Computing (Extended Abstract)”, Proceedings of the Financial Cryptography and Data Security, pp. 171-175. 2008.

[16] D. Basin, S. Mödersheim, y L. Viganò, “An On-the-Fly Model-Checker for Security Protocol Analysis”, Proceedings of the 8th European Symposium on Research in Computer Security, pp. 253-270. 2003.

[17] M. Baugher et al., “The Secure Real-time Transport Protocol (SRTP)”, Request for Comments (RFC) 3711, Internet Society, 2004.

[18] A. Becker, “Electronic Commerce: Concepts, Methodologies, Tools and Applications”, Information Science Reference, 2007.

[19] D. Beckett, “RDF/XML Syntax Specification (Revised). W3C Recommendation 10 February 2004”, W3C, Febrero 2004.

[20] G. Bella et al., “Formal Verification of Cardholder Registration in SET”, Proceedings of the 6th European Symposium on Research in Computer Security, pp. 159-174. 2004.

[21] G. Bella et al., “Making Sense of Specifications: The Formalization of SET (Transcript of Discussion)”, Revised Papers from the 8th International Workshop on Security Protocols, pp. 82-86. 2001.

[22] G. Bella, L.C. Paulson, y F. Massacci, “The verification of an industrial payment protocol: the SET purchase phase”, Proceedings of the 9th ACM conference on Computer and communications security, pp. 12-20. 2002.

[23] G. Bella, F. Massacci, y L.C. Paulson, “An overview of the verification of SET”, International Journal of Information Security, vol. 4, pp. 2005.

[24] M. Bellare et al., “VarietyCash: a multi-purpose electronic payment system”, Proceedings of the 3rd conference on USENIX Workshop on Electronic Commerce - Volume 3, pp. 2-2. 1998.

[25] T. Berners-Lee et al. “Uniform Resource Identifiers (URI): Generic Syntax”, RFC 3986, Enero 2005.

[26] H. Beykirch et. al., ”Payment API for v1.0 Internet Open Trading Protocol (IOTP)”, IETF Internet Draft. TRADE Working Group. September 2000.

[27] K. Birman, R.V. Renesse, y W. Vogels, “Adding High Availability and Autonomic Behavior to Web Services”, Proceedings of the 26th International Conference on Software Engineering, pp. 17-26. 2004.

[28] Y. Boichut et al., “Improvements on the Genet and Klay Technique to Automatically Verify Security Protocols”, Proceedings of Automated Verification of Infinite-State Systems (AVIS) 2004. 2004.

[29] J. Bosak, T. McGrath, y G.K. Holman, “Universal Business Language v2.0. Standard, 12 December 2006”, http://docs.oasis-open.org/ubl/os-UBL-2.0/UBL-2.0.html. Accedido: 19 de Diciembre de 2.008.

[30] S. Brands, “Untraceable off-line cash in wallet with observers”, Proceedings of the 13th annual international cryptology conference on Advances in cryptology, pp. 302-318. 1994.

[31] S. Brlek, S. Hamadou, y J. Mullins, “A flaw in the electronic commerce protocol SET”, Information Processing Letters, vol. 97, no. 3, pp. 104-108. 2006.

[32] D. Burdett, “Internet Open Trading Protocol - IOTP Version 1.0”, RFC 2801, 2000.

[33] D. Burdett, y M. Goncalves, Internet Open Trading Protocol, McGraw-Hill Professional, 2000.

[34] J. Camenisch, “Efficient Anonymous Fingerprinting with Group Signatures”. Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology, pp. 415-428. 2000.

[35] L.J. Camp, M. Sirbu, y J.D. Tygar, “Token and notational money in electronic commerce”, Proceedings of the 1st conference on USENIX Workshop on Electronic Commerce - Volume 1. 1995.

[36] O. Cánovas, y A. F. Gómez. “A Distributed Credential Management System for SPKI-Based Delegation Systems”, Proceedings of 1st Annual PKI Research Workshop, pp. 65–76, 2002.

[37] O. Cánovas, A.F. Gómez, y G. Martínez. "PISCIS: Comercio Electrónico basado en Infraestructuras de Certificación Avanzadas y Sistemas de Tarjeta Inteligente", Actas del I Simposio Nacional sobre Negocio Electrónico, pp. 201-216. 2001.

[38] O. Cánovas, A. F. Gómez, y **A. Ruiz-Martínez**, “Un sistema de fidelización electrónica basado en el uso de certificados de credencial”, Actas del II Simposio Español de Comercio Electrónico, pp. 37-48. 2003.

[39] S. Cantor et al., “Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0”, OASIS Standard, 15 Marzo 2005.

[40] G. Carat, “ePayment Systems database – Trends & Analysis – “. Electronic Payment System Observatory (ePSO). Institute for Prospective Technological Studies. 2002.

[41] M. Carbonell et al., “Secure e-payment protocol with new involved entities”, Proceedings of International Symposium on Collaborative Technologies and Systems, 2008 (CTS 2008.), pp. 103-111. 2008.

[42] M. Carbonell, J. M. Sierra, y J. López, “Secure multiparty payment with an intermediary entity”, Computers and Security, Vol. 28, no. 5, pp. 289-300. 2009.

[43] Cartio.com. http://cartio.com/. Accedido: 28 de Noviembre de 2.008.

[44] CEN/TC224/WG10: Inter-sector Electronic Purse, Part 2: Security Architecture, 1996.

[45] CEPSCO. CEPSCO LLC: Common Electronic Purse Specifications, March 1999.

[46] C. Chang, y Y. Lai, “A flexible date-attachment scheme on e-cash”, Computers & Security, vol. 22, no. 2, pp. 160-166. 2003.

[47] C.H. Chan, C. Cheng, y C. Hsu, “Bargaining strategy formulation with CRM for an e-commerce agent”, Electronic Commerce Research and Applications, vol. 6, no. 4, pp. 490-498. 2007.

[48] P. Chan et al., “Smart Card Payment over Internet with Privacy Protection”, Smart Card Research and Applications, pp. 98-104. 2000.

[49] D. Chaum, "Blind signatures for untraceable payments", Proceedings of Advances in Cryptology, CRYPTO'82, pp. 199-203, 1982.

[50] D. Chaum, “Security without identification: transaction systems to make big brother obsolete”, Commun. ACM, vol. 28, no. 10, pp. 1030-1044, 1985.

[51] Q. Chen, C. Zhang, y S. Zhang, “Verifying the Payment Authorization in SET Protocol”, Intelligent Data Engineering and Automated Learning, pp. 914-918. 2003.

[52] Z. Chen, Java Card(TM) Technology for Smart Cards: Architecture and Programmer's Guide, Prentice Hall PTR, 2000.

[53] Y. Chevalier et al., “A High Level Protocol Specification Language for Industrial Security-Sensitive Protocols”, Proceedings of Workshop on Specification and Automated Processing of Security Requirements (SAPS 2004), pp. 276-291. 2004.

[54] R. Chinnici et al., “Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language”, W3C Recommendation 26 June 2007, http://www.w3.org/TR/wsdl20/. Accedido: 19 de Diciembre de 2008.

[55] J. Choi, K. Sakurai, y J. Park, “Does It Need Trusted Third Party? Design of Buyer-Seller Watermarking Protocol without Trusted Third Party”, Proceedings of Applied Cryptography and Network Security 2003, pp. 265-279. 2003.

[56] E. Chung, y D. Dardailler. Joint Electronic Payment Initiative (JEPI). W3C note. 1997.

[57] L. Clement et al., UDDI Version 3.0.2. OASIS Standard, 2005.

[58] CodaLogic, “LMX™ W3C XML Schema to C++ Data Binding“, http://www.codalogic.com/lmx/?xmlfwd-xml2cppcom. Accedido: 28 de Noviembre de 2.008.

[59] Comisión del Mercado de las Telecomunicaciones, “Informe sobre el comercio electrónico en España a través de entidades de medios de pago (IV trimestre 2008)”. 2009.

[60] CommerceNet, http://www.commerce.net/. Accedido: 11 de Noviembre de 2.008.

[61] D. Cooper et. al, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile”, RFC 5280. 2008.

[62] N. Courtois, M. Finiasz, y N. Sendrier, “How to Achieve a McEliece-Based Digital Signature Scheme”, Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology, pp. 157-174. 2001.

[63] B. Cox, J.D. Tygar, y M. Sirbu, “NetBill security and transaction protocol,” Proceedings of the 1st conference on USENIX Workshop on Electronic Commerce - Volume 1, 1995.

[64] L. Cranor et al, “The Platform for Privacy Preferences 1.1 (P3P1.1) Specification”. W3C. Noviembre 2006.

[65] X. Dai, y J. Grundy, “NetPay - an efficient protocol for micro-payments on the WWW”, Proceedings of the Fifth Australian World Wide Web Conference, 1999.

[66] X. Dai y J. Grundy, “Three Kinds of E-wallets for a NetPay Micro-Payment System”, Proceedings of Web Information Systems – WISE 2004, pp. 66-77. 2004.

[67] X. Dai, y J. Grundy, “NetPay: An off-line, decentralized micro-payment system for thin-client applications”, Electron. Commer. Rec. Appl., vol. 6, no.1, pp. 91-101. 2007.

[68] S. Darko-Ampem, M. Katsoufi, y P. Giambiagi, “Secure Negotiation in Virtual Organizations”, Proceedings of the 10th IEEE on International Enterprise Distributed Object Computing Conference Workshops, pp. 48-54. 2006.

[69] A. Das, “Payment agents”, Payment technologies for E-commerce, Springer-Verlag New York, Inc., pp. 149-170, 2003.

[70] K. Davidson, y Y. Kawatsura, “Digital Signatures for the v1.0 Internet Open Trading Protocol (IOTP)”, RFC 2802, 2000.

[71] C. de Laat, G. Gross, L. Gommas, y J. Vollvrecht, “Generic AAA Architecture”, Request For Comments (RFC) 2903, Internet Society, 2000.

[72] S. Devane, M. Chatterjee, y D. Phatak, “Secure E-Commerce Protocol for Purchase of e-Goods - Using Smart Card”, Proceedings of the Third International Symposium on Information Assurance and Security, IEEE Computer Society, pp. 9-14. 2007.

[73] T. Dierks, ”The TLS Protocol Version 1.0”, RFC 2246, January 1999.

[74] D. Dolev, y A.C. Yao, “On the security of public key protocols”, 22nd Annual Symposium on Foundations of Computer Science (SFCS 1981), pp. 350-357. 1981.

[75] J. Domingo, A. Martínez-Ballesté, y F. Sebé, “Vídeos de pago en Internet”, Boletines de Rediris, nº 62-63, , pp. 6-9. Diciembre 2002-enero 2003

[76] J. Domingo-Ferrer, y A. Martínez-Ballesté, “STREAMOBILE: Pay-per-View Video Streaming to Mobile Devices Over the Internet,” Proceedings of the 13th International Workshop on Database and Expert Systems Applications, IEEE Computer Society, pp. 418-422. 2002.

[77] G.N. Drew, “Using Set for Secure Electronic Commerce”, Prentice Hall PTR, 1998.

[78] Z. Đurić, O. Marić, y D. Gašević, “Internet Payment System: A New Payment System for Internet Transactions”, Journal of Universal Computer Science, vol. 13, pp. 479-503. 2007.

[79] D. Eastlake 3rd, S. Crocker, J. Schiller, ”Randomness Recommendations for Security”, RFC 1750, December 1994.

[80] D. E. Eastlake 3rd, “Universal Payment Preamble”, Internet-Draft, August 1996.

[81] D. Eastlake 3rd, "Internet Open Trading Protocol Version 2 Requirements", RFC 3354, 2002.

[82] D. Eastlake 3rd, "Internet Open Trading Protocol (IOTP) Version 1, Errata", RFC 3504, 2003.

[83] D. Eastlake 3rd, J. Reagle, y D. Solo, "(Extensible Markup Language) XML-Signature Syntax and Processing", RFC. 3275. 2002.

[84] D. Eastlake, J. Reagle, y D. Solo, (editors), "XML-Signature Syntax and Processing: W3C Recommendation: 12 February 2002", Febrero 2002; http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/. Accedido: 11 Noviembre 2008.

[85] D. Eastlake et al. (editors), "XML Signature Syntax and Processing (Second Edition). W3C Recommendation 10 June 2008.", Nov. 2008; http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/. Accedido 11 Noviembre 2008.

[86] J. Elien, "Certificate Discovery Using SPKI/SDSI 2.0 Certificates", Master's Thesis, 1998.

[87] C. Ellison et al., "SPKI Certificate Theory", Request for Comments (RFC) 2693. Sep. 1999.

[88] S. El Sawda, P. Urien, y I. Hajjeh, "Non Repudiation for SIP Protocol; SIP Sign", Proceedings of ICTTA 2008. 3rd International Conference on Information and Communication Technologies: From Theory to Applications, 2008, pp. 1-5. 2008.

[89] EMVCo, LLC, "EMV 2000 Integrated Circuit Card Specification for Payment Systems Version 4.1 – Book 1: Application Independent IC Card to Terminal Interface Requirements", 2004.

[90] EMVCo, LLC, "EMV 2000 Integrated Circuit Card Specification for Payment Systems Version 4.1 – Book 2: Security and Key Management," 2004.

[91] EMVCo, LLC, "EMV 2000 Integrated Circuit Card Specification for Payment Systems Version 4.1 – Book 3: Application Specification", 2004.

[92] EMVCo, LLC, "EMV 2000 Integrated Circuit Card Specification for Payment Systems Version 4.1 – Book 4: CardHolder, Attendant, and Acquirer Interface Requirements", 2004.

[93] P. England y T. Tariq, "Towards a Programmable TPM", Proceedings of TRUST, L. Chen, C.J. Mitchell, y A. Martin, eds., Springer, pp. 1-13. 2009.

[94] T. Erl, "Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services", Prentice Hall PTR, 2004.

[95] European Committee for Standardization, "Secure signature-creation devices EAL 4+ in CEN Workshop Agreement (CWA) 14169", European Committee for Standardization. 2004.

[96] European Parliament, "Directive 1999/93/EC of the European Parliament and the council of December 1999 on a Community framework for electronic signatures. Official Journal of the European Communities". 2000.

[97] P.D. Ezhilchelvan, y S.K. Shrivastava, “A Family of Trusted Third Party Based Fair-Exchange Protocols”, IEEE Trans. Dependable Secur. Comput., vol. 2, no. 4, pp. 273-286. 2005.

[98] D. C. Fallside, y P. Walmsley, “XML Schema Part 0: Primer Second Edition. W3C Recommendation 28 October 2004”, Octubre 2004.

[99] C. -I. Fan, W.-K. Chen, y Y.-S. Yeh, “Date attachable electronic cash”, Computer Communications, vol. 23, no. 4, pp. 425-428. Feb. 2000.

[100] C. Fan, Y. Liang, y B. Lin, “Fair Transaction Protocols Based on Electronic Cash”, Proceedings of the Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies, IEEE Computer Society, pp. 383-388. 2006.

[101] J. Farrel, y H. Lausen, “Semantic Annotations for WSDL and XML Schema“. W3C Working Draft 10 April 2007. http://www.w3.org/TR/2007/WD-sawsdl-20070410/. Accedido: 17 de Noviembre de 2.008.

[102] D. Fensel, “Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce”, Springer, 2001.

[103] J. Fischl, y H. Tschofenig, “Making SIP make cents”, Queue, vol. 5, no. 2, pp. 42-49. 2007.

[104] W. Ford y M.S. Baum, Secure Electronic Commerce: Building the Infrastructure for Digital Signatures and Encryption, Prentice Hall, 1997.

[105] Y. Frankel, Y. Tsiounis, y M. Yung, “Indirect Discourse Proof: Achieving Efficient Fair Off-Line E-cash”, Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security: Advances in Cryptology, Lecture Notes in Computer Science, pp. 286-300. 1996.

[106] F. Frattolillo y F. Landolfi, “Designing a DRM System”, Proceedings of International Symposium on Information Assurance and Security, pp. 221-226. 2008.

[107] A. O. Freier, P. Karlton, y P. C. Kocher, “The SSL protocol version 3.0. Internet draft”, Netscape Communications, November 1996. http://www.mozilla.org/projects/security/pki/nss/ssl/draft302.txt. Accedido: 3 de Marzo de 2.009.

[108] M. Fritscher, y O. Kump, “Security And Productivity Improvements – Sufficient For The Success Of Secure Electronic Transaction?”, Proceedings of the 8th European, pp. 909-913. 2000.

[109] K. Fujimura, y D. Eastlake, “Requirements and Design for Voucher Trading System (VTS)”, RFC 3506, 2003.

[110] E. Gallery, y C. Mitchell, “Trusted Mobile Platforms”, Foundations of Security Analysis and Design IV, pp. 282-323. 2007.

[111] A. Ganesh, y K. Gopinath, “SPKI/SDSI certificate chain discovery with generic constraints”, Proceedings of the 1st Bangalore annual Computer conference, ACM, pp. 1-8. 2008.

[112] J.A. Garay, M. Jakobsson, y P.D. MacKenzie, "Abuse-Free Optimistic Contract Signing", Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology, Lecture Notes in Computer Science, pp. 449-466. 1999.

[113] F. J. García, A. F. Gómez, G. López, R. Marín y **A. Ruiz-Martínez**, "PISCIS: E-commerce based on an advanced certification infrastructure and smart cards", Revista Upgrade, Vol. III, nº 6, pp. 16-21. Diciembre 2002.

[114] F. J. García, A. F. Gómez, G. López, R. Marín y **A. Ruiz-Martínez**, "PISCIS: Comercio electrónico basado en una infraestructura de certificación avanzada y tarjetas inteligentes", en Monográfico "Seguridad en e-Comercio", Revista Novática nº 160, pp. 6-11. Noviembre-Diciembre 2002.

[115] M. A. García Lax, "SESTERTIUM: Arquitectura segura y escalable para servidores de pago con monedero electrónico e integración en Web", Jornadas Técnicas Rediris 2002.

[116] P. Garner, R. Edwards, y P. Coulton, "Card-based Macropayment for Mobile Phones", Proceedings of the International Conference on Mobile Business, 2006.

[117] S. Glassman et al., "The Millicent protocol for inexpensive electronic commerce", World Wide Web Journal, pp. 603-618. 1995.

[118] D. Gefen, E. Karahanna, y D. Straub, "Inexperience and experience with online stores: the importance of TAM and trust," Engineering Management, IEEE Transactions on, vol. 50, pp. 307-321. 2003.

[119] M. Gudgin et al., "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition) W3C Recommendation 27 April 2007", W3C, Abril 2007. http://www.w3.org/TR/soap12. Accedido: 11 de Noviembre de 2.008.

[120] S. Gürgens, C. Rudolph, y H. Vogt, "On the security of fair non-repudiation protocols", Int. J. Inf. Secur., vol. 4, no. 4, pp. 253-262. 2005.

[121] A. Hamzehei, "Digital Rights Management using RFID in an E-Commerce Environment", World Applied Sciences Journal, vol. 5, no. 3, pp. 324-331. 2008.

[122] G. Hanaoka, Y. Zheng, y H. Imai, "LITESET: A light-weight secure electronic transaction protocol", Proceedings of Information Security and Privacy, pp. 215-226. 1998.

[123] M. Handley, V. Jacobson, y C. Perkins, "SDP: Session Description Protocol", Request For Comments (RFC) 4566, Internet Society, 2006.

[124] U. Hansmann et al., "Smart Card Application Development Using Java", Springer, 2002.

[125] J. Hao, J. Zou, y Y. Dai, "A Real-Time Payment Scheme for SIP Service Based on Hash Chain", Proceedings of IEEE International Conference on E-Business Engineering, pp. 279-286. 2008.

[126] M. Hassinen, K. Hyppönen, y K. Haataja, "An Open, PKI-Based Mobile Payment System", Proceedings of Emerging Trends in Information and Communication Security, pp. 86-100. 2006.

[127] M. Hassinen, K. Hyppönen, y E. Trichina, "Utilizing national public-key infrastructure in mobile payment systems", Electron. Commer. Rec. Appl., vol. 7, no. 2, pp. 214-231. 2008.

[128] M. He, N. R. Jennings, y H. Leung, "On Agent-Mediated Electronic Commerce", IEEE Transactions on Knowledge and Data Engineering, vol. 15, no. 4, pp. 985-1003. 2003.

[129] J. L. Hernández-Ardieta, A. I. González-Tablas, y B. Ramos Álvarez, "An optimistic fair exchange protocol based on signature policies", Computers & Security, vol. 27, no. 7-8, pp. 309-322. 2008.

[130] E.V. Herreweghen y U. Wille, "Risks and potentials of using EMV for internet payments", Proceedings of the USENIX Workshop on Smartcard Technology on USENIX Workshop on Smartcard Technology, USENIX Association, pp. 18-18. 1999.

[131] E.V. Herreweghen, "Non-repudiation in SET: Open Issues," Proceedings of the 4th International Conference on Financial Cryptography, Lecture Notes in Computer Science, pp. 140-156. 2001.

[132] A. Hezberg, "Micropayments", Payment technologies for E-commerce, Springer-Verlag. 2003.

[133] K. E. B. Hickman, "The SSL protocol. Internet draft", Netscape Communications, February 1995. http://www.mozilla.org/projects/security/pki/nss/ssl/draft02.html. Accedido: 3 de Marzo de 2.009.

[134] D. Hou, y H. Xia, "Application of Streaming Media Technology in Electronic Commerce", Proceedings of 2008 International Symposium on Electronic Commerce and Security, pp. 850-853. 2008.

[135] S. Ho, R.J. Kauffman, y T. Liang, "A growth theory perspective on B2C e-commerce growth in Europe: An exploratory study", Electronic Commerce Research and Applications, vol. 6, no. 3, pp. 237-259. 2007.

[136] R. Housley, "Cryptographic Message Syntax (CMS)", RFC 3852, Julio 2004.

[137] M. Huang, H. Chen, y W. Lee, "An Efficient Non-repudiation Mechanism for SIP-Based Services", Proceedings of International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2008 (IIHMSP '08), pp. 1541-1544. 2008.

[138] M. S. Hwang, I. C. Lin, and L. H. LI, "A simple Micro-payment scheme", The Journal of Systems and Software vol. 55, no. 3, pp. 221-229, 2001.

[139] M. Hwang y P. Sung, "A Study of Micro-payment Based on One-way Hash Chain", International Journal of Network Security, vol. 2, no. 2, pp. 81-90. 2006.

[140] J. Hwang, T. Yeh, y J. Li, "Securing on-line credit card payments without disclosing privacy information", Computer Standards & Interfaces, vol. 25, no. 2, pp. 119-129. 2003.

[141] International Organisation for Standardisation (ISO), "ISO 4217: Codes for the Representation of Currencies", 1995.

[142] International Organisation for Standardisation (ISO), JTC 1/SC 17, "ISO/IEC 7816 Identification cards - Integrated circuit(s) cards with contacts", 1987.

[143] I. Ion y B. Dragovic, "Don't trust POS terminals! Verify in-shop payments with your phone", Proceedings of SMPU'08 Workshop on Security and Privacy Issues in Mobile Phone Use. Pervasive 2008. 2008.

[144] ITU-T, "ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)". ITU Recommendation X.690 (ISO/IEC 8825-1:2002). 2002.

[145] P. Jarupunphol and C.J. Mitchell, "Measuring SSL and SET against e-commerce consumer requirements", Proceedings of International Network Conference 2002 (INC 2002), Plymouth University Press, pp. 323-330. 2002.

[146] C. Jennings et al., "Payment for Services in Session Initiation Protocol (SIP)", Internet-Draft, Internet Society, 2008.

[147] M. Jiang, y A. Willey, "Service-Oriented Architecture for Deploying and Integrating Enterprise Applications", Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture, IEEE Computer Society, pp. 272-273. 2005.

[148] W. Juang, "D-cash: A flexible pre-paid e-cash scheme for date-attachment", Electronic Commerce Research and Applications, vol. 6, no. 1, pp. 74-80. 2007.

[149] J. Jürjens, "Modelling audit security for Smart-Card payment schemes with UML-SEC", Proceedings of the 16th international conference on Information security: Trusted information: the new decade challenge, pp. 93-107. 2001.

[150] J. Jürjens, y G. Wimmel, "Security Modelling for Electronic Commerce: The Common Electronic Purse Specifications", Proceedings of the IFIP Conference on Towards The E-Society: E-Commerce, E-Business, E-Government, Kluwer, B.V., pp. 489-506. 2001.

[151] C. S. Jutla and M. Yung, "PayTree: Amortized-signature for flexible micro-payments", Proceedings of Second USENIX Association Workshop on Electronic Commerce, pp. 213-221. 1996.

[152] R. Kalakota, y M. Robinson, "E-Business: Roadmap for Success", Addison-Wesley, 1999.

[153] Y. Kawakura et al., "Flexible and Scalable Credential Structures: NetBill Implementation and Experience", Proceedings of the International Workshop on Cryptographic Techniques and E-Commerce, pp. 231-245. 1999.

[154] N. Kew, "The Apache Modules Book: Application Development with Apache", Prentice Hall PTR, 2007.

[155] I. Kim et al., "Model-Based Analysis of Money Accountability in Electronic Purses", Internet and Network Economics, 2005.

[156] S. Kim and W. Lee, "A PayWord-based Micro-payment protocol supporting multiple payments", Proceedings of the 12th International Conference on Computer Communications and Networks (ICCCN 2003), pp. 609-612. 2003.

[157] H. Knospe, y S. Schwiderski-Grosche, “Secure mobile commerce”, C. J. Mitchell, editor, Security for Mobility, IEE Press, pp. 325-346. 2004.

[158] D. Koo, “The fundamental reasons of e-consumers’ loyalty to an online store”, Electronic Commerce Research and Applications, vol. 5, no. 2, pp. 117-130. 2006

[159] J. Kopecký et al., “SAWSDL: Semantic Annotations for WSDL and XML Schema”, IEEE Internet Computing, vol. 11, no. 6, pp. 60-67. 2007.

[160] W. Kou, “Payment Technologies for E-Commerce”, Springer, 2003.

[161] S.H. Kwok et al., “Integration of digital rights management into the internet open trading protocol”, Decision Support Systems., vol. 34, no. 4, pp. 413-425. 2003.

[162] R. Kowalczyk, M. Ulieru, y R. Unland, “Integrating Mobile and Intelligent Agents in Advanced E-commerce: A Survey”, Agent Technologies, Infrastructures, Tools, and Applications for E-Services, pp. 295-313. 2003.

[163] S. Kremer, O. Markowitch, y J. Zhou, “An intensive survey of fair non-repudiation protocols”, Computer Communications, vol. 25, no. 17, pp. 1606-1621. 2002.

[164] K.O. Kürtz et al., “Selecting theories and nonce generation for recursive protocols”, Proceedings of the 2007 ACM workshop on Formal methods in security engineering, ACM, pp. 61-70. 2007.

[165] G. Lacoste et al., “SEMPER - Secure Electronic Marketplace for Europe”, Springer, 2000.

[166] G. Lacoste et al., “Chapter 11: The Payment Framework”, SEMPER - Secure Electronic Marketplace for Europe, 2000.

[167] C. Lei, P. Yu, P. Tsai, y M. Chan, “An efficient and anonymous buyer-seller watermarking protocol”, IEEE Transactions on Image Processing, vol. 13, no. 12, pp. 1618-1626. 2004.

[168] M. Lesk, “Micropayments: An Idea Whose Time Has Passed Twice?”, IEEE Security and Privacy, vol. 2, no. 1, pp. 61-63. 2004.

[169] A. Levi y C. Koç, “CONSEPP: CONvenient and Secure Electronic Payment Protocol Based on X9.59,” Proceedings of the 17th Annual Computer Security Applications Conference, IEEE Computer Society, pp. 286. 2001.

[170] X. Liang, Z. Cao, R. Lu, y L. Qin, “Efficient and secure protocol in fair document exchange”, Comput. Stand. Interfaces, vol. 30, no. 3, pp. 167-176. 2008.

[171] H. Liaw, J. Lin, y W. Wu, “A new electronic traveler's check scheme based on one-way hash function”, Electronic Commerce Research and Applications, vol. 6, no. 4, pp. 499-508. 2007.

[172] B. Limthanmaphon, Y. Zhang, y Z. Zhang, “An Agent-Based Negotiation Model Supporting Transactions in Electronic Commerce,” Proceedings of the 11th International Workshop on Database and Expert Systems Applications, IEEE Computer Society, pp. 440. 2000.

[173] K. Lin, “E-Commerce Technology: Back to a Prominent Future”, IEEE Internet Computing, vol. 12, no. 1, pp. 60-65. 2008.

[174] M. Liu et al., "Security Mechanism Research of EMV2000", Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops, IEEE Computer Society, pp. 307-310. 2007.

[175] Y. Liu y Z. Zhao, "Payment Scheme for Multi-Party Cascading P2P Exchange", Integration and Innovation Orient to E-Society, volume 1, pp. 560-567. 2008.

[176] S. Lu, y S.A. Smolka, "Model Checking the Secure Electronic Transaction (SET) Protocol", Proceedings of 7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, vol. 311, pp. 358-365. 1999.

[177] H. Ludwing et als. "Web Service Level Agreement (WSLA) Language Specification", IBM Corporation. 2003.

[178] J. Luo et al., "Adding OWL-S Support to the Existing UDDI Infrastructure", Proceedings of the IEEE International Conference on Web Services, IEEE Computer Society, pp. 153-162. 2006.

[179] D.C. Lynch, y L. Lundquist, "Digital Money: The New Era of Internet Commerce", John Wiley & Sons, 1995.

[180] J. Luo et al., "Adding OWL-S Support to the Existing UDDI Infrastructure", Proceedings of the IEEE International Conference on Web Services, IEEE Computer Society, pp. 153-162. 2006.

[181] P. Maes, R.H. Guttman, y A.G. Moukas, "Agents that buy and sell", Commun. ACM, vol. 42, no. 3, pp. 81-100. 1999.

[182] A. Mankin et al., "Change Process for the Session Initiation Protocol (SIP)", Request For Comments (RFC) 3427, Internet Society, 2002.

[183] O. Markowitch y S. Kremer, "A Multi-party Optimistic Non-repudiation Protocol", Proceedings of the Third International Conference on Information Security and Cryptology, Springer-Verlag, pp. 109-122. 2001.

[184] A. Martínez-Ballesté, J. Domingo-Ferrer, y F. Sebé, "MINPAY: a Multi-device INternet PAY-as-you-watch system", Proceedings of the International Conference on Information Technology: Computers and Communications, IEEE Computer Society, pp. 258-262. 2003.

[185] R. Martínez Perea, "Internet Multimedia Communications Using SIP: A Modern Approach Including Java Practice", Morgan Kaufmann, 2008.

[186] M.T. Masonta, O.J. Oyedapo, y A.M. Kurien, "Mobile Client for the Next Generation Networks", Proceedings of Broadband Communications, Information Technology & Biomedical Applications, pp. 274-279. 2008.

[187] K. Mayes, y K. Markantonakis, "Smart Cards, Tokens, Security and Applications", Springer, 2008.

[188] G. Medvinsky, y C. Neuman, "NetCash: a design for practical electronic currency on the Internet", Proceedings of the 1st ACM conference on Computer and communications security, Fairfax, Virginia, United States: ACM, pp. 102-106. 1993.

[189] N. Memon y P. Wong, “A buyer-seller watermarking protocol”, IEEE Transactions on Image Processing, vol. 10, no. 4, pp. 643-649. 2001.

[190] B. Meng y H. Zhang, “An Electronic Commerce System Prototype and Its Implementations”, Proceedings of the The Fifth International Conference on Computer and Information Technology, IEEE Computer Society, pp. 966-970. 2005.

[191] M.S. Merkow, J. Breithaupt, y K.L. Wheeler, “Building SET applications for secure transactions”, John Wiley & Sons, Inc., 1998.

[192] S. Micali y R.L. Rivest, “Micropayments Revisited”, Proceedings of the The Cryptographer's Track at the RSA Conference on Topics in Cryptology, Lecture Notes in Computer Science, pp. 149-163. 2002.

[193] T. Michel, “Common Markup for micropayment per-fee-links”, W3C, 1999.

[194] Microsoft Developer Network, “Asynchronous Pluggable Protocols”, MSDN Library, http://msdn.microsoft.com/en-us/library/aa767743(VS.85).aspx, Accedido: 28 de Noviembre de 2.008.

[195] Microsoft Developer Network, “Cryptographic Service Providers”, http://msdn.microsoft.com/en-us/library/aa380245(VS.85).aspx. Accedido: 3 de Abril de 2.009.

[196] J.C. Mitchell, V. Shmatikov, y U. Stern, “Finite-state analysis of SSL 3.0”, Proceedings of the 7th conference on USENIX Security Symposium, 1998 - Volume 7, USENIX Association, pp. 16-22. 1998.

[197] S. Mohammadi y S. Abedi, “ECC-Based Biometric Signature: A New Approach in Electronic Banking Security”, Proceedings of 2008 International Symposium on Electronic Commerce and Security, pp. 763-766. 2008.

[198] Mozilla, “Mozilla Developper Documentation”, http://www.mozilla.org/docs/. Accedido: 11 de Noviembre de 2.008.

[199] D.P. Mundy, y D.D. Chadwick, “Comparing the Performance of Abstract Syntax Notation One (ASN.1) vs eXtensible Markup Language (XML)”, Proceedings Terena Networking Conference, 2003.

[200] A. Nadalin et. al. “Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)”, OASIS Standard Specification. Febrero 2006.

[201] National Institute of Standards and Technology (NIST), “Secure Hash Standard, FIPS PUB 180-2”, 2002.

[202] B.C. Neuman, y G. Medvinsky, “Requirements for network payment: the NetCheque perspective”, Proceedings of the 40th IEEE Computer Society International Conference, IEEE Computer Society, pp. 32-42. 1995.

[203] NewGenPay, http://www.newgenpay.com/. Accedido: 11 de Noviembre de 2.008.

[204] H. Nielsen et al., “PEP - an Extension Mechanism for HTTP,” Nov. 1997; http://www.w3.org/TR/WD-http-pep-971121. Accedido: 11 de Noviembre de 2.008.

[205] K. Ogata, y K. Futatsugi, “Equational Approach to Formal Analysis of TLS”, Proceedings of International Conference on Distributed Computing Systems, pp. 795-804. 2005.

[206] S. Oh, “A Stakeholder Perspective on Successful Electronic Payment Systems Diffusion”, Proceedings of the 39th Annual Hawaii International Conference on System Sciences - Volume 08, IEEE Computer Society, pp. 186-172. 2006.

[207] D. O'Mahony, M.A. Peirce, y H. Tewari, Electronic Payment Systems, Artech House Publishers, 1997.

[208] K. Ogata y K. Futatsugi, “Formal Analysis of the NetBill Electronic Commerce Protocol”, Software Security - Theories and Systems, pp. 45-64. 2004.

[209] J.A. Onieva et al., “Intermediary Non-repudiation Protocols”, Proceedings of 2003 IEEE International Conference on E-Commerce Technology (CEC'03), pp. 207-.214 2003.

[210] J.A. Onieva, J. Zhou, y J. Lopez, “Multiparty nonrepudiation: A survey”, ACM Computing Survey, vol. 41, pp. 1-43, 2008.

[211] OpenSSL Project. http://www.openssl.org/. Accedido: 28 de Noviembre de 2008.

[212] G. Pacifi, “Performance management for cluster-based Web services”, IEEE Journal on Selected Areas in Communications, vol. 23, no. 12, pp. 2333- 2343. 2005.

[213] M. P. Papazoglou et al., “Service-Oriented Computing: State of the Art and Research Challenges”, Computer, vol. 40, no. 11, pp. 38-45. 2007.

[214] V. Pasupathinathan et al, “Formal analysis of card-based payment systems in mobile devices”, Proceedings of the 2006 Australasian workshops on Grid computing and e-research - Volume 54, Australian Computer Society, Inc., pp. 213-220. 2006.

[215] V. Patil, y R. Shyamasundar, “e-coupons: An Efficient, Secure and Delegable Micro-Payment System”, Information Systems Frontiers, vol. 7, no. 28-31, pp. 371-389. 2005.

[216] L.C. Paulson, “Inductive analysis of the Internet protocol TLS”, ACM Trans. Inf. Syst. Secur., vol. 2, no. 3, pp. 332-351. 1999.

[217] Paypal. https://www.paypal.com. Accedido: 9 de Marzo de 2009.

[218] J.M. Peha y I.M. Khamitov, “PayCash: a secure efficient Internet payment system”, Proceedings of the 5th international conference on Electronic commerce, ACM, pp. 125-130. 2003.

[219] J.M. Peha y I.M. Khamitov, “PayCash: a secure efficient internet payment system”, Electronic Commerce Research and Applications, vol. 3, pp. 381-388. 2004.

[220] J.L.A. Peiro et al., “Designing a generic payment service”, IBM Syst. J., vol. 37, no. 1, pp. 72-88. 1997.

[221] R.M. Perea, “Internet Multimedia Communications Using SIP: A Modern Approach Including Java Practice”, Morgan Kaufmann, 2008.

[222] B. Pfitzmann, y M. Schunter, “Asymmetric Fingerprinting”, Proceedings of Eurocrypt’97, Konstanz, Germany. Springer-Verlag, pp. 84-95. 1996.

[223] B. Pfitzmann y M. Waidner, "Anonymous fingerprinting", Proceedings of Eurocrypt'97, Berlín, Lecture Notes in Computer Science, pp. 84–95. 1997.

[224] C. Popescu, "An off-line electronic cash system with revokable anonymity", Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference, 2004 (MELECON 2004), pp. 763-767. 2004.

[225] C. Popescu, y H. Oros, "An off-line electronic cash system with anonymity revoking trustee", Proceedings of the International on Theory and Applications of Mathematics and Informatics, pp. 409-416. 2004.

[226] C. Popescu, "A Fair Off-line Electronic Cash System Based on Elliptic Curve Discrete Logarithm Problem", Studies in Informatics and Control, vol. 14, no. 4, pp. 291-298. 2005.

[227] K. Pousttchi, y D. G. Wiedemann, "Charging of mobile services by mobile payment reference model", Handbook E-Money, E-Payment & M-Payment, pp. 363-377. 2005.

[228] T. Poutanen, H. Hinton, y M. Stumm, "NetCents: a lightweight protocol for secure micropayments", Proceedings of the 3rd conference on USENIX Workshop on Electronic Commerce - Volume 3, USENIX Association, pp. 3-10. 1998.

[229] A. Prasad, V. Mahajan, y B. Bronnenberg, "Advertising versus pay-per-view in electronic media", International Journal of Research in Marketing, vol. 20, no. 1, pp. 13-30. 2003.

[230] M.O. Rabin, "Digitalized Signatures And Public-Key Functions As Intractable As Factorization", Massachusetts Institute of Technology, 1979.

[231] C. Radu, "Implementing Electronic Card Payment Systems", Artech House Publishers, 2002.

[232] N. Ragouzis Et al, "Security assertion markup language v2.0 technical overview", Technical report, OASIS Security Services (SAML) TC, 2007.

[233] J. Raja, M. Senthil velmurgan, y A. Seetharaman, "E-payments: Problems and Prospects", Journal of Internet Banking and Commerce, vol. 13, no. 1, pp. 1-17. 2008.

[234] W. Rankl, y W. Effing, "Smart Card Handbook", Wiley, 2004.

[235] F. F. Reichheld, "Loyalty-based management", Harvard Business Review, vol. 71, no. 2, pp. 64-73.

[236] E. Rescorla, "SSL and TLS: Designing and Building Secure Systems", Addison-Wesley Professional, 2000.

[237] R.L. Rivest, "Peppercoin Micropayments", Proceedings of Financial Cryptography, Lecture Notes in Computer Science, pp. 2-8. 2004.

[238] R.L. Rivest, y A. Shamir, "PayWord and MicroMint: Two Simple Micropayment Schemes", Proceedings of the International Workshop on Security Protocols, Lecture Notes in Computer Science, pp. 69-87. 1997.

[239] A.W. Röhm, G. Pernul, y G. Herrmann, "Modeling Secure and Fair Electronic Commerce", Proceedings of the 14th Annual Computer Security Applications Conference, IEEE Computer Society, pp. 155-160. 1998

[240] J. Rosenberg et al., "SIP: Session Initiation Protocol", Request For Comments (RFC) 3261, Internet Society, 2002.

[241] J. Rosenberg y H. Schulzrinne, "Reliability of Provisional Responses in the Session Initiation Protocol (SIP)", Request For Comments (RFC) 3262, Internet Society, 2002.

[242] J. Rosenberg, y H. Schulzrinne, "An Offer/Answer Model with the Session Description Protocol (SDP)", Request For Comments (RFC) 3264, Internet Society, 2002.

[243] H. Rossnagel, "Mobile Qualified Electronic Signatures and Certification on Demand", Proceedings of Public Key Infrastructure, pp. 274-286. 2004.

[244] H. Rossnagel, y D. Royer, "Making Money with Mobile Qualified Electronic Signatures", Proceedings of Trust, Privacy and Security in Digital Business, pp. 110-118. 2005.

[245] RSA Laboratories, "PKCS #7: Cryptographic Message Syntax Standard. RSA Laboratories Technical Note. Versión 1.5", Noviembre, 1993.

[246] RSA Laboratories, "PKCS #10: Certification Request Syntax Standard", http://www.rsa.com/rsalabs/node.asp?id=2132. Accedido: 3 de Abril de 2.009.

[247] RSA Laboratories, "PKCS #11: Cryptographic Token Interface Standard", http://www.rsa.com/rsalabs/node.asp?id=2133. Accedido: 3 de Abril de 2.009.

[248] **A. Ruiz-Martínez**, C.I. Marín-López, L. Baño-López, y A.F. Gómez-Skarmeta, "A new fair non-repudiation protocol for secure negotiation and contract signing", Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services, ACM, págs. 1-11, 2006.

[249] **A. Ruiz-Martínez**, G. Martínez, Ó. Cánovas y A. F. Gómez-Skarmeta, "SPEED Protocol: Smartcard-Based Payment with Encrypted Electronic Delivery", Information Security Conference, pp. 446-461. 2001.

[250] **A. Ruiz-Martínez**, L. Baño, C. I. Marín, Ó. Cánovas, A. F. Gómez-Skarmeta, "A SECURe infrastructure for negotiation and purchase of Intellectual Property Rights (SECURingIPR)". Proceedings de IASTED International Conference on Communication, Network, and Information Security (CNIS'03), pp. 277-282. 2003.

[251] **A. Ruiz-Martínez**, Ó. Cánovas y A.F. Gómez-Skarmeta, "Combination of a Smartcard E-Purse and E-Coin to Make Electronic Payments on the Internet", SECRYPT, pp. 203-206. 2006.

[252] **A. Ruiz-Martínez**, Ó. Cánovas, y A. F. Gómez-Skarmeta, "Smartcard-Based e-Coin for Electronic Payments on the (Mobile) Internet," SITIS, IEEE Computer Society, pp. 361-368. 2007.

[253] **A. Ruiz-Martínez**, Ó. Cánovas, y A.F. Gómez-Skarmeta, "Towards a generic per-fee-link framework", 2nd International Conference on Digital Information Management (ICDIM'07), IEEE Computer Society, pp. 37-42. 2007.

[254] **A. Ruiz-Martínez**, Ó.Cánovas, y A.F. Gómez-Skarmeta, "Payment Annotations for Web Service Description Language (PA-WSDL)," Computer and Information Science, pp. 65-76. 2008.

[255] **A. Ruiz-Martínez**, Ó. Cánovas, A. F. Gómez-Skarmeta. "Un framework genérico para el soporte de pagos por clic". Actas de la X Reunión Española sobre Criptología y Seguridad de la Información (RECSI). 2008.

[256] **A. Ruiz-Martínez**, Ó.Cánovas, y A.F. Gómez-Skarmeta, "Design and implementation of a generic per-fee-link framework", Internet Research. 2009. (Aceptado).

[257] **A. Ruiz-Martínez**, D. Sánchez-Martínez, M. Martínez-Montesinos, y A.F. Gómez-Skarmeta, "A survey of electronic signature solutions in mobile devices", J. Theor. Appl. Electron. Commer. Res., vol. 2, no. 3, pp. 94-109. 2007.

[258] **A. Ruiz-Martínez**, D. Sánchez-Martínez, M. Martínez-Montesinos, y A.F. Gómez-Skarmeta, "Mobile Signature Solutions for Guaranteeing Non-Repudiation in Mobile Business and Mobile Commerce," Mobile and Ubiquitous Commerce: Advanced E-Business Methods, Information Science Reference, pp. 116-135. 2009.

[259] **A. Ruiz-Martínez**, J.A. Sánchez-Laguna, y A.F. Gómez-Skarmeta, "A Payment Framework for Internet Media-on-Demand Services based on RTSP", Proceedings of 4th IEEE Consumer Communications and Networking Conference, 2007 (CCNC 2007), IEEE Computer Society, pp. 965-970. 2007.

[260] **A. Ruiz-Martínez**, J.A. Sánchez-Laguna, y A.F. Gómez-Skarmeta, "SIP extensions to support (micro)payments", Proceedings of 21st IEEE International Conference On Advanced Information Networking And Applications (AINA2007), IEEE Computer Society, pp. 289-296. 2007.

[261] **A. Ruiz-Martínez**, J. A. Sánchez-Laguna, y A. F. Gómez-Skarmeta, "Towards a Generic Payment Framework for Internet Media-on-Demand Services Based on RTSP", Proceedings of 22nd IEEE International Conference On Advanced Information Networking And Applications (AINA2008), IEEE Computer Society, pp. 826-831. 2008

[262] M. C. Ruiz et al., "Analysis of the SET e-commerce protocol using a true concurrency process algebra", Proceedings of the 2006 ACM symposium on Applied computing, ACM, pp. 879-886. 2006.

[263] P.M. Ruiz et al., "Multimedia Control Protocols for Wireless Networks", Emerging Wireless Multimedia, N.P. Apostolis K. Salkintzis, ed., pp. 83-120. 2005.

[264] A. Sadeghi, "Trusted Computing – Special Aspects and Challenges", Proceedings of SOFSEM 2008: Theory and Practice of Computer Science, pp. 98-117. 2008.

[265] N. Sadeh, "M-Commerce: Technologies, Services, and Business Models", Wiley, 2002.

[266] J. Sahut, “Internet Payment Solutions: Comparative Evaluation and Key Factors of Success”, Proceedings of the 2005 Symposium on Applications and the Internet Workshops, 2005. Saint Workshops 2005, pp. 354-357. 2005.

[267] P. Sandoz et al., “Fast Web Services”, technical report; 2003, disponible en: http://developer.java.sun.com/developer/technicalArticles/WebServices/fastWS/ (accedido 29 Julio 2008).

[268] C. Schmidt, y R. Müller, “A framework for micropayment evaluation”, NETNOMICS, vol. 1, Oct. 1999, pp. 187-200.

[269] B. Schoenmakers, “Security Aspects of the Ecash™ Payment System”, State of the Art in Applied Cryptography, Lecture Notes in Computer Science, pp. 338-352. 1998.

[270] H. Schulzrinne, A. Rao, y R. Lanphier, “Real Time Streaming Protocol (RTSP)”, Request For Comments (RFC) 2326, Internet Society, 1998.

[271] H. Schulzrinne et al., “RTP: A Transport Protocol for Real-Time Applications”, Request For Comments (RFC) 3550, Internet Society, 2003.

[272] S. Schwiderski-Grosche, y H. Knospe, “Secure Mobile Commerce”,,C. Mitchell (editor): Special issue of the IEE Electronics and Communication Engineering Journal on Security for Mobility, vol. 14, no. 5, pp. 228-238, 2002.

[273] SEMPER Project. http://www.semper.org. Accedido: 10 de Noviembre de 2.008.

[274] SETCo. “Secure Electronic Transaction Specification – Books 1–4”, SETCo, 1997.

[275] R. Shang, Y. Chen, y L. Shen, “Extrinsic versus intrinsic motivations for consumers to shop on-line”, Information & Management, vol. 42, pp. 401-413. 2005.

[276] C. Shannon, “Communication theory of secrecy systems”, Bell System Technical Journal, vol. 28, no. 3, pp. 656-715. 1949.

[277] J. Sahut, “Internet Payment Solutions: Comparative Evaluation and Key Factors of Success”, Proceedings of the 2005 Symposium on Applications and the Internet Workshops, pp. 354-357. 2005.

[278] G. Shen y X. Zheng, “Research on Implementation of Elliptic Curve Cryptosystem in E-Commerce”, Proceedings of 2008 International Symposium on Electronic Commerce and Security, pp. 288-291, 2008.

[279] M.H. Sherif, “Protocols for Secure Electronic Commerce”, Second Edition, CRC, 2003.

[280] M.A. Sirbu, y J.C. Chuang, “Distributed Authentication in Kerberos Using Public Key Cryptography”, Proceedings of the 1997 Symposium on Network and Distributed System Security, IEEE Computer Society, pp. 134-139. 1997.

[281] H. Soo Ju, H. Jeong Kim, D. Hoon Lee, y J. In Lim, “An Anonymous Buyer-Seller Watermarking Protocol with Anonymity Control”, Proceedings of Information Security and Cryptology – ICISC 2002, pp. 421-432. 2002.

[282] W. Stallings, “Cryptography and Network Security”, Prentice Hall, 2005.

[283] Sun Microsystems, "Java Card Technology", http://java.sun.com/javacard/. Accedido: 27 de Marzo de 2.009.

[284] Sun Microsystems, "Security and Trust Services API for J2ME (SATSA); JSR 177", http://java.sun.com/products/satsa/. Accedido: 3 de Abril de 2.009.

[285] J. Talbot, y D. Welsh, "Complexity and Cryptography: An Introduction", Cambridge University Press, 2006.

[286] M. Terada, y K. Fujimura, "Voucher Trading System Application Programming Interface (VTS-API)", Request For Comments (RFC) 4154, Internet Society, 2005.

[287] Y. Tian, y C. Stewart, "History of E-Commerce", Electronic Commerce: Concepts, Methodologies, Tools and Applications, Information Science Reference, 2007.

[288] M. Tian et al., "Performance Impact of Web Services on Internet Servers", Proceedings of IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2003). 2003.

[289] M. Toorani, y A. Beheshti Shirazi, "SSMS - A secure SMS messaging protocol for the m-payment systems", Proceedings of IEEE Symposium on Computers and Communications, 2008. ISCC 2008, pp. 700-705. 2008.

[290] G.W. Treese, y L.C. Stewart, "Designing Systems for Internet Commerce", Addison-Wesley Professional, 1998.

[291] M. Tripunitara, y T. Messerges, "Resolving the Micropayment Problem", Computer, vol. 40, no. 2, pp. 104-106. 2007.

[292] A. Tsalgatidou, y T. Pilioura, "An Overview of Standards and Related Technology in Web Services", Distributed and Parallel Databases, vol. 12, pp. 135-162. 2002.

[293] T. Tsiakis y G. Sthephanides, "The concept of security and trust in electronic payments", Computers & Security, vol. 24, pp. 10-15. 2005.

[294] M. Turuani, "Sécurité des Protocoles Cryptographiques: Décidabilité et Complexité, Phd", Université Henri Poincaré, 2003.

[295] J.D. Tygar, "Atomicity in electronic commerce", Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing, ACM, pp. 8-26. 1996.

[296] J.D. Tygar, "Atomicity versus Anonymity: Distributed Transactions for Electronic Commerce", Proceedings of the 24rd International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc., pp. 1-12. 1998.

[297] S. Ueda et al., "Authenticating Video Streams", Proceedings of the 20th International Conference on Advanced Information Networking and Applications - Volume 1 (AINA'06) - Volume 01, IEEE Computer Society, pp. 863-868. 2006.

[298] United Nations, "UNCITRAL Model Law on Electronic Signatures with Guide to Enactment". United Nations Publications. 2002.

[299] U.S. Census Bureau, "Quarterly Retail E-Commerce Sales", http://www.census.gov/mrts/www/ecomm.html, Accedido: 10 de Marzo de 2009.

[300] Visa Europe, "3D SET", http://www.visaeu.com/. Accedido: 5 de Marzo de 2009.

[301] Visa Partner Network, "Visa 3-D Secure Specifications", https://partnernetwork.visa.com/vpn/global/category.do?categoryId=88&documentId=127&userRegion=1. Accedido: 5 de Marzo de 2.009.

[302] K. Verma, y A. Sheth, "Semantically Annotating a Web Service", IEEE Internet Computing, vol. 11, no. 2, pp. 83-85. 2007.

[303] L. Viganò, "Automated Security Protocol Analysis With the AVISPA Tool", Electronic Notes in Theoretical Computer Science, Proceedings of the 21st Annual Conference on Mathematical Foundations of Programming Semantics (MFPS XXI), vol. 155, pp. 61-86. 2006.

[304] D. Wagner y B. Schneier, "Analysis of the SSL 3.0 protocol", Proceedings of the Second UNIX Workshop on Electronic Commerce, pp. 29-40. 1996.

[305] G. Wang et al., "Generic, Optimistic, and Efficient Schemes for Fair Certified Email Delivery", Proceedings of Information and Communications Security, 2005.

[306] H. Wang, H. Guo, y J. Yin, "A Fair Item-Item Exchange Protocol Satisfying Newly Introduced Requirements", Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05) Volume 2 - Volume 02, IEEE Computer Society, pp. 502-507. 2005.

[307] S. Wang, F. Hong, y G. Cui, "A fair e-cash payment scheme based on credit", Proceedings of the 7th international conference on Electronic commerce, ACM, pp. 622-626. 2005.

[308] Y. Wang, y T. Li, "LITESET/A++: A New Agent-Assisted Secure Payment Protocol", Proceedings of the IEEE International Conference on E-Commerce Technology, IEEE Computer Society, pp. 244-251. 2004.

[309] J. Wareham, J. G Zheng, y D. Straub, "Critical themes in electronic commerce research: a meta-analysis", Journal of Information Technology, vol. 20, pp. 1-19. 2005.

[310] P. Wayner, "Digital Cash: Commerce on the Net", Academic Press, 1997.

[311] R.Weber, "Chablis - Market Analysis of Digital Payment Systems", Technical Report, 1998.

[312] Word Wide Web Consortium (W3C). "XML-Signature Syntax and Processing. W3C Recommendation. 12 February 2002". 2002.

[313] Word Wide Web Consortium (W3C). "XML-Encryption Syntax and Processing. W3C Recommendation. 12 February 2002". 2002.

[314] Word Wide Web Consortium (W3C), "ODRL, W3C Note", 2002. Disponible en: http://www.w3.org/TR/odrl/ accedido: 2 de Agosto de 2.008.

[315] Y. Wu y H. Pang, "A light weight buyer-seller watermarking protocol", Adv. MultiMedia, vol. 2008, pp. 1-7. 2008.

[316] X. Yi et al., "A Secure Agent-based Framework for Internet Trading in Mobile Computing Environments", Distributed and Parallel Databases, vol. 8, no. 1, pp. 85-117. 2000.

[317] H. Yu, K. Hsi, y P. Kuo, "Electronic payment systems: an analysis and comparison of types", Technology in Society, vol. 24, no. 3, pp. 331-347. 2002.

[318] X. Zeng, J. Zeng, y Q. Guo, "Research of trust on B2C electronic commerce", Proceedings of the 7th international conference on Electronic commerce, ACM, pp. 221-225. 2005

[319] G. Zhang, F. Cheng, y C. Meinel, "SIMPA: A SIP-Based Mobile Payment Architecture", Proceedings of the Seventh IEEE/ACIS International Conference on Computer and Information Science (ICIS 2008), IEEE Computer Society, pp. 287-292. 2008.

[320] G. Zhang, F. Cheng, y C. Meinel, "Towards Secure Mobile Payment Based on SIP", Proceedings of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2008), IEEE Computer Society, pp. 96-104, 2008.

[321] Q. Zhang, K. Markantonakis, y K. Mayes, "A Practical Fair-Exchange E-Payment Protocol for Anonymous Purchase and Physical Delivery", Proceedings of the IEEE International Conference on Computer Systems and Applications (AICCSA), IEEE Computer Society, pp. 851-858. 2006

[322] W. Zhang y R.V. Engelen, "A Table-Driven Streaming XML Parsing Methodology for High-Performance Web Services", Proceedings of the IEEE International Conference on Web Services, IEEE Computer Society, pp. 197-204. 2006.

[323] Y. Zhang y X. Liu, "Running-mode analysis of the Security Socket Layer protocol", ACM SIGOPS Operating Systems Review vol. 38, no. 2, pp. 34-40. 2004.

[324] J. Zhou, "Non-Repudiation in Electronic Commerce", Artech House Publishers, 2001.

[325] P. Zimmermann, A. Johnston y J. Callas, "ZRTP: Media Path Key Agreement for Secure RTP", Internet Draft draft-zimmermann-avt-zrtp-11, Internet Society, 2008.

[326] Zongkai, L. Weimin, y T. Yunmeng, "A New Fair Micropayment System Based on Hash Chain", Proceedings of the 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'04), IEEE Computer Society, pp. 139-145. 2004.

[327] V. Zwass, "Electronic commerce: structures and issues", Int. J. Electron. Commerce, vol. 1, no. 1, pp. 3-23. 1996.

Apéndice A
Acrónimos

AAA	Authentication, Authorization and Accouting – Autenticación, Autorización y Auditoría
AF	Asymmetric Fingerprinting – Fingerprinting asimétrico
APDU	Application Protocol Data Unit – Unidad de Datos del Protocolo de Aplicación
API	Application Programming Interface – Interfaz de Programación de Aplicaciones
ASN.1	Abstract Syntax Notation One – Notación de Sintaxis Abracta Uno
AVISPA	Automated Validation of Internet Security Protocols and Applications – Validación Automatizada de Aplicaciones y Protocolos de Seguridad en Internet
BSW	Buyer-Seller Watermarking – Watermarking Comprador-Vendedor
CA	Certification Authority – Autoridad de Certificación
CEN	Comité Européen de Normalisation – Comité Europeo de Normalización
CEPS	Common Electronic Purse Specifications – Especificaciones Comunes de Monedero Electrónico
CMS	Cryptographic Message Syntax – Sintaxis de Mensajes Criptográficos
CRL	Certificate Revocation List – Lista de Certificados Revocados
CSP	Cryptographic Service Provider – Proveedor de Servicios Criptográficos
DCMS	Distributed Credential Management System – Sistema de Gestión de Credenciales Distribuido
DRM	Digital Rights Language – Lenguaje de Derechos Digitales
DSK	Derived Symmetric Key – Clave Simétrica Derivada
EMV	Europay, MasterCard, Visa

EPP	Extended Payment Protocol – Protocolo de Pago Extendido
ETSI	European Telecommunications Standards Institute – Instituto Europeo de Estándares de Telecomunicaciones
HLPSL	High Level Protocol Specification Language – Lenguaje de Especificación de Protocolos de Alto Nivel
HTTP	HyperText Transfer Protocol – Protocolo de Transferencia de HiperTexto
IOTP	Internet Open Trading Protocol – Protocolo de Negocio Abierto para Internet
ISC	Internet Systems Consortium – Consorcio de Sistemas de Internet
ISO	International Organization for Standardization – Organización Internacional para la Estandarización
ISP	Internet Service Provider – Proveedor de Servicios de Internet
J2ME	Java 2 Micro Edition – actualmente conocida como Java ME (Java Mobile Edition).
JEPI	Joint Electronic Payment Initiative – Iniciativa de Pago Electrónica Conjunta
MP3	MPEG-1 Audio Layer 3
OCSP	Online Certificate Status Protocol – Protocolo de Comprobación de Certificados on-line.
ODRL	Open Digital Rights Language – Lenguaje de Derechos Digitales Abierto
PA-WSDL	Payment Annotations for WSL – Anotaciones de Pago para WSDL
PDA	Personal Digital Assistant – Asistente Digital Personal
PIN	Personal Identification Number – Número de Identificación Personal
PISCIS	Proyecto Piloto de definición de una Infraestructura de Seguridad para el Comercio Inteligente de Servicios
PKCS	Public-Key Cryptography Standards – Estándares de Criptografía Pública
PKI	Public Key Infrastructure – Infraestructura de Clave Pública
PRACK	Provisional Response Acknowledgement – Confirmación de

	Respuesta Provisional
PSP	Payment Service Provider – Proveedor de Servicios de Pago
RFC	Request For Comments – Petición de Comentarios
RTP	Real Time Protocol – Protocolo de Tiempo Real
RTSP	Real Time Streaming Protocol – Protocolo de Streaming en Tiempo Real
	Security Assertion Markup Language – Lenguaje de Marcado de Sentencias de Seguridad
SAM	Secure Application Module – Módulo de Aplicación Segura
SATSA	Security and Trust Services API for J2ME – API de Servicios de Seguridad y Confianza para J2ME
SDP	Session Description Protocol – Protocolo de Descripción de Sesiones
SDSI	Simple Distributed Security Infrastructure – Infraestructura Sencilla de Seguridad Distribuida
SECuringIPR	SECure infrastructure for negotiation and purchase of Intellectual Property Rights – Infraestructura de Seguridad para la Negociación y Compra de Derechos de Propiedad Intelectual
SEMPER	Secure Electronic Marketplace for Europe – Mercado Electrónico Seguro para Europa
SET	Secure Electronic Transaction – Transacción Electrónica Segura
SLA	Service Level Agreement – Acuerdo a Nivel de Servicio
SIM	Subscriber Identity Module – Módulo de Identidad del Suscriptor
SIP	Session Initiation Protocol – Protocolo de Inicio de Sesión
SOAP	Simple Object Access Protocol – Protocolo Sencillo de Acceso a Objetos
SPEED	SmartCard-based Protocol with Encrypted Electronic Delivery – Protocolo de Pago basado en Tarjeta Inteligente con Entrega Electrónica Cifrada
SPKI	Simple Public Key Infrastructure – Infraestructura Sencilla de Clave Pública
SRTP	Secure Real-time Transport Protocol – Protocolo de Transporte Seguro en Tiempo Real

SSCD	Secure Signature Creation Device – Dispositivo Seguro de Creación de Firma
TPM	Trusted Platform Module – Módulo de Plataforma Confiable
TTP	Trusted Third Party – Tercero de Confianza
UDDI	Universal Description, Discovery and Integration – Integración, Descubrimeinto y Descripción Universal
VoIP	Voice over IP – Voz sobre IP
WG10	CEN EN 1546 standard (Intersector Electronic Purse) also known as WG10.
WS	Web Service – Servicio Web
WSDL	Web Service Description Language – Lenguaje de Descripción de Servicios Web
WSLA	Web Service Level Agreeements – Acuerdos a Nivel de Servicio Web
XML	Extensible Markup Language – Lenguaje de Etiquetas Extensible
ZRTP	Media Path Key Agreement for Secure RTP – Acuerdo de claves para Tráfico Multimedia para el RTP seguro

Apéndice B
Especificación HLPSL de SPEED

```
%SPEED Protocol

role speed_client( A, B, T : agent,
                   Pka, Pks : public_key,
                   Snd, Rcv : channel(dy),
                   Snd2, Rcv2 : channel(dy))
played_by A
def=

  local State, SeqN                 : nat,
      NID, ProductID, Price         : text,
      Product, AccountNumber, Poid  : text,
      PaymentMode                   : text,
      Flag                          : bool,
      Enkey, Signkey, K, Reckey     : symmetric_key,
      Pkb, Pkt                      : public_key

  init State := 0

  transition

   1.       State = 0
            /\ Rcv(start)
            =|>
            State' := 1
            /\ NID' := new() /\ SeqN':=new() /\ ProductID':=new() /\
            Price':=new() /\ Enkey':=new() /\ Signkey':=new()
            /\ Flag':=new()
            /\
Snd({Pka.A.{Pka.A}_inv(Pks).NID'.SeqN'.ProductID'.Price'.Enkey'.Signkey
'.B.Flag'.{A.NID'.SeqN'.ProductID'.Price'.Enkey'.Signkey'.B.Flag'}_inv(
Pka)}_Pkb)
            /\
witness(A,B,negreq,{A.NID'.SeqN'.ProductID'.Price'.Enkey'.Signkey'.B.Fl
ag'}_inv(Pka))
      /\ secret(Enkey',enkeyk,{A,B})
      /\ secret(Signkey',signkeyk,{A,B})
```

```
   3.       State = 3
            /\
Rcv({B.A.NID'.SeqN'.Price'.Flag'.h(B.A.Signkey'.NID'.SeqN'.Price'.Signk
ey'.Flag')}_Enkey')
            =|>
            State' := 5
      /\
      Snd({Pka.A.{Pka.A}_inv(Pks).{B.A.NID'.Price'}_inv(Pka)}_Enkey')
      /\ witness(A,B,handshake,{B.A.NID'.Price'}_inv(Pka))

   5. State = 5
      /\
Rcv({Product'}_K'.{Pkb.B.{Pkb.B}_inv(Pks).B.A.{AccountNumber'.K'}_Pkt.h
(Product').h(ProductID).Poid'.Price'.Reckey'.{B.A.{AccountNumber'.K'}_P
kt.h(Product').h(ProductID).Poid'.Price'.Reckey'}_inv(Pkb)}_Enkey')
      =|>
      State' := 7
      /\
request(B,A,sbill,{B.A.{AccountNumber'.K'}_Pkt.h(Product').h(ProductID)
.Poid'.Price'.Reckey'}_inv(Pkb))
      /\ PaymentMode' :=new()
      /\
Snd({{Pkb.B.{Pkb.B}_inv(Pks).A.T.{AccountNumber'.K'}_Pkt.h(Product').h(
ProductID).Poid'.Price'.Reckey'.{A.T.{AccountNumber'.K'}_Pkt.h(Product'
).h(ProductID).Poid'.Price'.Reckey'}_inv(Pkb).PaymentMode'}_inv(Pka)}_P
kt)
      /\
witness(A,T,sbillc,{Pkb.B.{Pkb.B}_inv(Pks).A.T.{AccountNumber'.K'}_Pkt.
h(Product').h(ProductID).Poid'.Price'.Reckey'.{A.T.{AccountNumber'.K'}_
Pkt.h(Product').h(ProductID).Poid'.Price'.Reckey'}_inv(Pkb).PaymentMode
'}_inv(Pka))

    7.  State = 7
      /\
Rcv2({Pkt.T.{Pkt.T}_inv(Pks).C.B.Poid'.Price'.K'.{T.A.B.Poid'.Price'.K'
}_inv(Pkb)}_Reckey)
      =|> State' := 8
      /\ request(A,T,receipt,{T.A.B.Poid'.Price'.K'}_inv(Pkb))

end role
%-----------------------------------------------------------------------
role speed_vendor (B, A, T     : agent,
```

```
                  Pkb, Pks     : public_key,
                  Snd, Rcv     : channel(dy),
                  Snd2, Rcv2   : channel(dy))
played_by B
def=

   local  State, SeqN                    : nat,
          Nb                             : text,
          NID, ProductID, Price          : text,
          Product, AccountNumber, Poid   : text,
          Flag                           : bool,
          Enkey, Signkey, K, Reckey      : symmetric_key,
          Pka, Pkt                       : public_key

   init State := 0

   transition

   1.       State = 0
            /\
Rcv({Pka'.A.{Pka'.A}_inv(Pks).NID'.SeqN'.ProductID'.Price'.Enkey'.Signk
ey'.B.Flag'.{A.NID'.SeqN'.ProductID'.Price'.Enkey'.Signkey'.B.Flag'}_in
v(Pka')}_Pkb)
            =|>
            State' := 2
            /\ Nb' := new()
            /\
Snd({B.A.NID'.SeqN'.Price'.Flag'.h(Signkey'.NID'.B.A.SeqN'.Price'.Flag'
.Signkey')}_Enkey')
            /\
request(B,A,negreq,{NID'.SeqN'.ProductID'.Price'.Enkey'.Signkey'.B.Flag
'}_inv(Pka'))
      /\    secret(Enkey',enkeyk,{A,B}) /\
      secret(Signkey',signkeyk,{A,B})

   2. State = 2
      /\
      Rcv({Pka'.A.{Pka'.A}_inv(Pks).{A.B.NID'.Price'}_inv(Pka)}_Enkey')
            =|>
      State' := 4
      /\ request(B,A,handshake,{A.B.NID'.Price'}_inv(Pka))
      /\ K' := new()
```

```
      /\ Product':=new()
      /\ Reckey':=new()
      /\ AccountNumber' :=new()
      /\ Poid' :=new()
      /\ AccountNumber' :=new()
      /\
Snd({Product'}_K'.{Pkb.B.{Pkb.B}_inv(Pks).B.A.{AccountNumber'.K'}_Pkt.h
(Product').h(ProductID).Poid'.Price'.Reckey'.{B.A.{AccountNumber'.K'}_P
kt.h(Product').h(ProductID).Poid'.Price'.Reckey'}_inv(Pkb)}_Enkey')
      /\
witness(B,A,sbill,{B.A.{AccountNumber'.K'}_Pkt.h(Product').h(ProductID)
.Poid'.Price'.Reckey'}_inv(Pkb))
      /\ secret(K',keyk,{A,B,T})
      /\ secret(Reckey',reckeyk,{A,B,T})

    4.      State = 4
      /\
Rcv2({Pkt.T.{Pkt.T}_inv(Pks).A.B.Poid'.Price'.K'.{T.A.B.Poid'.Price'.K'
}_inv(Pkb)}_Reckey)
      =|> State' := 8
      /\ request(B,T,receipt,{T.A.B.Poid'.Price'.K'}_inv(Pkb))

end role
%-----------------------------------------------------------------------
role speed_broker (T, A, B     : agent,
                   Pkt, Pks : public_key,
                   Snd, Rcv : channel(dy),
                   Snd2, Rcv2 : channel(dy))
played_by T
def=

   local  State, SeqN                    : nat,
          Nb                             : text,
          NID, ProductID, Price          : text,
          Product, AccountNumber, Poid   : text,
          PaymentMode                    : text,
          Flag                           : bool,
          Enkey, Signkey, K, Reckey      : symmetric_key,
          Pka, Pkb                       : public_key

   init State := 6
```

```
   transition

   1.       State = 6
      /\
Rcv({{Pkb.B.{Pkb.B}_inv(Pks).B.A.{AccountNumber'.K'}_Pkt.h(Product').h(
ProductID').Poid'.Price'.Reckey'.{B.A.{AccountNumber'.K'}_Pkt.h(Product
').h(ProductID').Poid'.Price'.Reckey'}_inv(Pkb).PaymentMode'}_inv(Pka)}
_Pkt)
      =|>
      State' := 7
      /\
request(B,A,sbillc,{Pkb.B.{Pkb.B}_inv(Pks).B.A.{AccountNumber'.K'}_Pkt.
h(Product').h(ProductID).Poid'.Price'.Reckey'.{B.A.{AccountNumber'.K'}_
Pkt.h(Product').h(ProductID).Poid'.Price'.Reckey'}_inv(Pkb).PaymentMode
'}_inv(Pka))
      /\
Snd({Pkt.T.{Pkt.T}_inv(Pks).A.B.Poid'.Price'.K'.{A.B.Poid'.Price'.K'}_i
nv(Pkb)}_Reckey')
      /\ witness(T,A,receipt,{A.B.Poid'.Price'.K'}_inv(Pkb))
      /\
Snd2({Pkt.T.{Pkt.T}_inv(Pks).A.B.Poid'.Price'.K'.{T.A.B.Poid'.Price'.K'
}_inv(Pkb)}_Reckey')
      /\ witness(T,B,receipt,{T.A.B.Poid'.Price'.K'}_inv(Pkb))

end role
%-----------------------------------------------------------------------
role session (A, B, T         : agent,
              Pka, Pkb, Pkt   : public_key,
              Pks             : public_key) def=

  local SA, RA, SB, RB, SAT, RAT, SBT, RBT: channel (dy)

  composition

           speed_client(A,B,T,Pka,Pks,SA,RA,SAT,RAT)
            /\ speed_vendor(B,A,T,Pkb,Pks,SB,RB,SBT,RBT)
            /\ speed_broker(T,A,B,Pkt,Pks,SAT,RAT,SBT,RBT)

end role
%-----------------------------------------------------------------------
role environment() def=

  const
```

```
        negreq, handshake, enkeyk, signkeyk    : protocol_id,
        keyk, reckeyk                          : protocol_id,
        sbill, sbillc, receipt                 : protocol_id,
        a, b, t                                : agent,
        pka, pkb, pkt, pks, pki                : public_key,
        h                                      : hash_func

  intruder_knowledge={a,b,t,pks,pki,inv(pki)}

  composition

        session(a,b,t,pka,pkb,pkt,pks)
     /\ session(a,b,t,pka,pkb,pkt,pks)
     /\ session(b,a,t,pkb,pka,pkt,pks)
     /\ session(a,i,t,pka,pki,pkt,pks)
     /\ session(i,b,t,pki,pkb,pkt,pks)

end role
%-----------------------------------------------------------------------
goal

      authentication_on negreq
      authentication_on handshake
      authentication_on sbill
      authentication_on sbillc
      authentication_on receipt
      secrecy_of enkeyk, signkeyk, keyk, reckeyk

end goal
%-----------------------------------------------------------------------
environment()
```

Apéndice C
Especificación ASN.1 de SPEED

```
SPEED-Protocol DEFINITIONS
::=
BEGIN


SPEEDMessage ::=
      CHOICE {
            negotiationRequest      [0] NegotiationRequest,
            productRequest          [1] ProductRequest,
            negotiationStep         [2] NegotiationStep,
            stopMegotiation         [3] StopNegotiation,
            handshake               [4] Handshake,
            cipheredProductDelivery [5] CipheredProductDelivery,
            paymentOrder            [6] PaymentOrder,
            samModuleMessage        [7] SAMModuleMessage,
            purseMessage            [8] PurseMessage,
            purchaseReceipt         [9] PurchaseReceipt,
            errorMessage            [10] ErrorMessage }


NegotiationRequest ::=
      ENVELOP{PublicKey,SIGNED{PrivateKey, NegotiationRequestContent}}

NegotiationRequestContent ::=
      SEQUENCE {
            cID PartyID,
            nid NID,
            seqN INTEGER,
            productID ProductID,
            price Price OPTIONAL,
            vendorID partyID,
            enKey SymmetricKey,
            signKey SymmetricKey,
            flag BOOLEAN,
            credentials Credentials OPTIONAL }
```

```
ProductRequest ::=
 ENVELOP{PublicKey, SIGNED{PrivateKey, ProductDeliveryRequestContent}}

ProductRequestContent ::=
      SEQUENCE {
            cID PartyID,
            nid NID,
            credentials Credentials OPTIONAL,
            productID ProductID,
            price Price,
            vendorID partyID,
            enKey SymmetricKey }


NegotiationStep ::=
      ENCRYPTED-SYMMETRIC{SymmetricKey,
            HMAC{SimplifiedContent, SymmetricKey}}


SimplifiedContent ::=
      SEQUENCE {
            cID PartyID,
            vID PartyID,
            nid NID,
            seqN INTEGER,
            price Price,
            flag BOOLEAN }

StopNegotiation ::=
      ENCRYPTED-SYMMETRIC{SymmetricKey,
            SIGNED{PrivateKey, StopNegotiationContent }}

StopNegotiationContent ::=
      SEQUENCE {
            cID PartyID,
            vID PartyID,
            nid NID,
            abort BOOLEAN,
            reason PrintableString OPTIONAL }
```

```
Handshake::=
  ENCRYPTED-SYMMETRIC{SymmetricKey,
                      SIGNED{PrivateKey, HandshakeContent}}

HandshakeContent ::=
      SEQUENCE {
            cID PartyID,
            vID PartyID,
            nid NID,
            price Price }

CipheredProductDelivery ::=
      SEQUENCE {
            cipheredProduct ENCRYPTED-SYMMETRIC{SymmetricKey, Product},
            content
                  CHOICE {
                        normal [0]
                              ENCRYPTED-SYMMETRIC{SymmetricKey,
                                    CipheredProductDeliveryContent},
                        fast  [1]
                        ENVELOPED{PublicKey,
                               CipheredProductDeliveryContent} }

CipheredProductDeliveryContent ::= SIGNED{PrivateKey, Bill}

Bill ::=
      SEQUENCE {
            cID PartyID,
            vID PartyID,
            toThirdPartyData ENVELOP{PublicKey, VendorSecret},
            cipheredProductHash
                  HASH{ENCRYPTED-SYMMETRIC{SymmetricKey, Product}},
            descriptionHash HASH{ProductID},
            paymentOrderID PaymentOrderID,
            dateTime DateTime,
            price Price,
            recKey SymmetricKey }
```

```
VendorSecret ::=
      SEQUENCE {
            AccountNumber PrintableString(SIZE(20)),
            paymentOrderID PaymentOrderID,
            symmetricKey SymmetricKey }


PaymentOrder ::=
      ENVELOP{PublicKey, SIGNED{PrivateKey, PaymentOrderContent}}


PaymentOrderContent ::=
      SEQUENCE {
            cID PartyID,
            bID PartyID,
            signedBill
                  SIGNED{PrivateKey, CipheredProductDeliveryContent},
            paymentMode PaymentMode }

PaymentMode ::=
      CHOICE {
            accountNumber AccountNumber,
            smartcardInf SEQUENCE {
                              bEncKey SymmetricKey,
                              bSingKey SymmetricKey,
                              serialNumber OCTET STRING,
                              transactionNumber INTEGER } }


SAMModuleAuthentication ::=
ENCRYPTED-SYMMETRIC{SymmetricKey,
                  HMAC{SAMMessageContent, SymmetricKey}}

SAMMessageContent ::=
      SEQUENCE {
            paymentOrderID paymentOrderID,
            s2 OCTET STRING }
```

```
PurseMessage ::=
      ENCRYPTED-SYMMETRIC{SymmetricKey,
                  HMAC{PurseMessageContent, SymmetricKey}}

PurseMessageContent ::=
      SEQUENCE {
            paymentOrderID paymentOrderID,
            s3 OCTET STRING }

PurchaseReceipt ::= ENCRYPTED-SYMMETRIC{SymmetricKey, receiptContent }

ReceiptContent ::= SIGNED{PrivateKey, Receipt}

Receipt ::=
      SEQUENCE {
            result BOOLEAN,
            paymentOrderID PaymentOrderID,
            price Price,
            symmetricKey SymmetricKey }

ErrorMessage ::= SIGNED{PrivateKey, ErrorMessageContent}

ErrorMessageContent ::=
      SEQUENCE {
            code NumericString(SIZE(3)),
            description PrintableString }

--- Basic Types

SymmetricKey ::= OCTET STRING
maxInt INTEGER ::= 4294967295
Credentials ::= PrintableString
Sign ::= PrintableString("+" | "-")
partyID ::= HASH{ PublicKey }

Price ::= SEQUENCE {
            quantity    NumericString (SIZE (6)),
            currency    PrintableString (SIZE (3)),
            sign-exp    Sign OPTIONAL,
```

```
            exp         NumericString(SIZE(6)) OPTIONAL }

NID ::= INTEGER (0 .. maxInt)       -- Negotiation IDentifier
Product ::= OCTET STRING

ProductID ::=
      SEQUENCE {
            class PrintableString(SIZE (3)),
            productNumber PrintableString(SIZE(6)),
            description PrintableString OPTIONAL }

PaymentOrderID ::=
      SEQUENCE {
            nid NID,
            cID PartyID,
            vID PartyID,
            cipheredProductHash
                  HASH{ENCRYPTED-SYMMETRIC{SymmetricKey, Product}} }

--- Macros and Operations

HASH { ToBeHashed }::=
      SEQUENCE {
            algorithmIdentifierHash AlgorithmIdentifierHash,
            hashtext BIT STRING }
(CONSTRAINED BY {
-- hashtext must be the results of the application of a
      valid digest procedure identified by --
-- -- algorithmIdentifierHash, --to -- ToBeHashed })

HMAC{ ToBeHashed, SymmetricKey }::=
      SEQUENCE {
            algorithmIdentifierHash AlgorithmIdentifierHash,
            keyIdentifier OCTET STRING,
            toBeHashed ToBeHashed,
                  hMac BIT STRING }
(CONSTRAINED BY {
-- hMac must be the results of the application of a valid HMAC
procedure identified by --
--  --algorithmIdentifierHash,  --to  --  ToBeHashed,  --  using  --
SymmetricKey })
```

```
ENCRYPTED-SYMMETRIC { SymmetricKey, ToBeEnciphered }::=
      SEQUENCE {
            algorithmIdentifierCipherSymmetric
                  AlgorithmIdentifierCipherSymmetric,
            keyIdentifier OCTET STRING,
            content BIT STRING }
(CONSTRAINED BY {
-- content must be the results of the application of a valid symmetric
encripted procedure, identified by --
-- -- algorithmIdentifierCipherSymmetric, -- to -- ToBeHashed, -- using
the symmetric key -- SymmetricKey })


SIGNED { PrivateKey, ToBeSigned } ::= --Like the PKCS#7 SignedData--

ENVELOP{PublicKey, ToBeEnciphered} ::= --Like the PKCS#7 EnvelopedData—

END
```

Apéndice D
Especificación HLPSL del proceso de certificación de PURSE-COIN

```
% --------------- ALICE = E-purse issuer -----------------------------

role alice (A, B : agent,
      Sne : text,
      Ka : public_key,
      Kab : symmetric_key,
      SND, RCV : channel(dy))
played_by A
def=
      local State : nat,
      Na : text,
      Kb : public_key

% Na es el challenge
% Kab es DSK
% Ka es la clave pública de Alice

const alice_bob_kb, alice_bob_na: protocol_id

init State := 0

transition

0. State = 0 /\ RCV(start) =|>
      State':= 2 /\ Na' := new()
      /\ SND(A.Na')

2. State = 2 /\ RCV(Sne'.{Na'.Kb'}_Kab) =|>
      State':= 4 /\ SND({Kb'}_inv(Ka))
      /\ witness(A,B,bob_alice_kb,{Sne.Kb'}_inv(Ka))
      /\ request(A,B,alice_bob_na,Na')
end role
```

```
% ------------------------ BOB = E-purse -----------------------------
role bob (A, B : agent,
      Sne : text,
      Ka : public_key,
      Kab : symmetric_key,
      SND, RCV : channel(dy))
played_by B
def=
      local State : nat,
      Nb,Na,N1b : text,
      Kb : public_key,
      K1ab : symmetric_key

const alice_bob_kb, alice_bob_na: protocol_id

% Na es el challenge
% Kab es DSK
% Ka es la clave pública del EPI
% Kb es la clave que genera bob para que sea certificada

init State := 1

transition

1. State = 1 /\ RCV(A.Na') =|>
      State':= 3 /\ Kb' := new()
      /\ SND(Sne.{Na'.Kb'}_Kab)
      /\ witness(B,A,alice_bob_na,Na')

3. State = 3 /\ RCV({Kb'}_inv(Ka)) =|>
      State':= 5
      /\ request(B,A,bob_alice_kb,{Sne.Kb'}_inv(Ka))

end role

%--------------------------- SESSION ----------------------------

role session(A, B : agent,
      Sne : text,
```

```
      Ka : public_key,
      Kab : symmetric_key)
def=

local SAB, RAB,
      SBA, RBA : channel (dy)

composition

      alice(A, B, Sne, Ka, Kab, SAB, RAB)
      /\ bob (A, B, Sne, Ka, Kab, SBA, RBA)

end role

role environment()
def=

const alice_bob_na, bob_alice_kb: protocol_id,
      a, b : agent,
      ka : public_key,
      kab, kai, kib : symmetric_key,
      ki: public_key,
      sne, snei : text

intruder_knowledge = {a, b, ka, kai, kib, ki, inv(ki)}

composition

      session(a,b,sne,ka,kab)
      /\ session(a,i,sne,ka,kai)

end role

goal

authentication_on bob_alice_nb
authentication_on alice_bob_na

end goal
```

```
environment()
```

Apéndice E
Especificación HLPSL del proceso de carga de PURSE-COIN

```
% ---------------------- ALICE = E-purse ------------------------------

role alice (A, B : agent,
      Sne : text,
      Ka : public_key,
      Kb : public_key,
      SND, RCV : channel(dy))
played_by A
def=
      local State : nat,
      Tne, Rnd, Am, Pay : text,
      Hash: hash_func,
      REQ, RES : message

% Ka es la clave pública del e-purse
% Kb es la clave pública del E-purse issuer

const bob_alice_req, alice_bob_res: protocol_id

init State := 0

transition

0. State = 0 /\ RCV(start) =|>
      State':= 2 /\ Tne' := new() /\ Rnd' := new() /\ Pay' := new()
      /\ REQ' := Pay'.{Hash(Kb).Sne.Tne'.Rnd'}_inv(Ka)
      /\ SND(REQ'.{A.Sne.Ka}_inv(Kb))
      /\ witness(A,B,bob_alice_req,REQ')

2. State = 2 /\ RCV(RES')
      /\ RES'={Hash(Ka).Sne.Tne'.Rnd'.Am'.Hash(Pay')}_inv(Kb) =|>
      request(A,B,alice_bob_res,RES')
end role
```

```
% ----------------- BOB = E-purse Issuer -----------------------------
role bob (A, B : agent,
      Sne : text,
      Ka : public_key,
      Kb : public_key,
      SND, RCV : channel(dy))
played_by B
def=
      local State : nat,
      Tne, Rnd, Am, Pay : text,
      K1ab : symmetric_key,
      Hash : hash_func,
      REQ, RES : message

const alice_bob_res, bob_alice_req: protocol_id

% Kb es la clave pública del EPI
% Kb es la clave que genera bob para que sea certificada
% am representa la cantidad a incrementar.

init State := 1

transition

1. State = 1 /\ RCV(REQ'.{Sne.Ka'}_inv(Kb))
      /\ REQ'=Pay'.{Hash(Kb).Sne.Tne'.Rnd'}_inv(Ka') =|>
      State':= 3
      /\ Am' := new()
      /\ RES' := {Hash(Ka).Sne.Tne'.Rnd'.Am'.Hash(Pay')}_inv(Kb)
      /\ SND(RES')
      /\ request(B,A,bob_alice_req,REQ')
      /\ witness(B,A,alice_bob_res,RES')

end role

%---------------------------- SESSION -----------------------------

role session(A, B : agent,
```

```
      Sne : text,
      Ka : public_key,
      Kb : public_key)
def=

local SAB, RAB,
      SBA, RBA : channel (dy)

composition

      alice(A, B, Sne, Ka, Ka, SAB, RAB)
      /\ bob (A, B, Sne, Ka, Ka, SBA, RBA)

end role

role environment()
def=

const bob_alice_req, alice_bob_res: protocol_id,
      a, b : agent,
      ka, kb, ki : public_key,
      sne, snei : text

intruder_knowledge = {a, b, ka, kb, ki, inv(ki)}

composition

      session(a,b,sne,ka,kb)
      /\ session(a,b,sne,ka,kb) % for checking replays

end role

goal

authentication_on bob_alice_req
authentication_on alice_bob_res

end goal

environment()
```

Apéndice F
Elementos XML para la descripción de la información de pago

En este apéndice describimos los distintos elementos XML que hemos definido para poder expresar la información de pago en los distintos frameworks de pagos presentados a lo largo de esta tesis.

En primer lugar, describiremos los elementos básicos, a continuación, explicaremos cómo estos elementos se pueden agrupar y referenciar de forma conjunta y, finalmente, presentaremos los elementos más importantes, es decir, los elementos que permiten describir toda la información de pago para un determinado producto.

Por cada uno de los elementos definidos hay un elemento XML que lo representa. Es importante mencionar que cada elemento definido posee un atributo denominado *ID* cuyo tipo es *ID* de los esquemas XML [98]. En la descripción de los diferentes elementos, este atributo ya no volverá a ser mencionado.

Para facilitar la explicación de estos elementos, nos basaremos en la página HTML que contiene un enlace de pago presentado en la Figura 4.7 para poner ejemplo de algunos de ellos.

Elementos Básicos

En este apartado describimos los elementos básicos que hemos definido para especificar información relacionada con el pago.

Elemento PaymentProtocol

El elemento *PaymentProtocol* contiene información sobre un protocolo de pago específico. Tiene un atributo llamado *ProtocolID* (*Identificador de Protocolo*) que identifica el protocolo de pago por medio de una URI [25]. Los principales elementos que contiene son: *Name* (*Nombre*), una cadena para nombrar el protocolo; *Version* (*Versión*), para especificar la versión del protocolo soportado; y, *AdditionalInformation* (*Información Adicional*), para incluir información adicional del protocolo. Ejemplos de protocolos de pago podrían ser Millicent, SPEED, así como cualquiera de los mencionados en el Capítulo 2.

En el ejemplo mencionado hemos definido un elemento *PaymentProtocol* para indicar que se soporta el protocolo de pago SPEED con identificador *ID1*.

Elemento Brand

El elemento *Brand* (*Marca financiera*) se define para proporcionar información acerca de una determinada marca financiera. Tiene un atributo denominado *BrandID*

(*Identificador de la Marca Financiera*) para identificar la marca financiera por medio de una URI [25]. La distinta información que podemos especificar sobre esta clase de marca es: *Name* (*Nombre*), el nombre de la marca financiera; *Description* (*Descripción*), una descripción de la marca; y, *Logo* (*Logotipo*) que es la URL que se puede utilizar para mostrar el logotipo de la marca financiera. Ejemplos de marcas financieras son: Mastercard, Visa, etc. En ejemplo de la Figura 4.7, la marca financiera es *Monedero Euro6000* y el elemento definido tiene el *ID2*.

Elemento PaymentServiceProvider

El elemento *PaymentServiceProvider* (*Proveedor de Servicios de Pago*) proporciona información con respecto al Proveedor de Servicios de Pago (PSP) que recibirá el pago. El PSP se identifica por medio de una URI en el atributo *PSPID*. Este rol puede ser representado por el propio vendedor o por una tercero de confianza para cliente y vendedor. Este elemento puede contener la siguiente información: *Name*, nombre del PSP; *Description* (*Descripción*), para especificar alguna información relacionado con el PSP; *Logo* (*Logotipo*), una URL al logotipo del PSP; *PaymentURL* (*URL de pago*), la URL donde enviar los mensajes de pago; *VendorID* (*Identificador del Vendedor*), identificador del vendedor en ese PSP, de forma que, esta información es utilizada para asociar el pago a este vendedor; y, *AdditionalData* (*Información Adicional*) que permite al vendedor proporcionar información adicional del pago al PSP. En este caso, el vendedor (*Research Papers Ltd*) desempeña este rol. En la Figura 4.7 aparece referenciado con el *ID3*.

Elemento Price

El elemento *Price* (*Precio*) se utiliza para describir la cantidad a pagar por el producto o servicio. Con este elemento podemos especificar: *Amount* (*Cantidad*), para especificar la cantidad a pagar; y, *Currency* (*Divisa*), para indicar la divisa en la que la cantidad está expresada. El formato seguido para expresar la divisa es conforme al estándar ISO 4217 [141]. En el ejemplo que estamos describiendo de la Figura 4.7, con el *ID4* aparece expresado la cantidad de 15 Euros.

Elemento PaymentModel

El elemento *PaymentModel* (*Modelo de Pago*) se ha definido para especificar el modelo de pago. Con este elemento indicamos si el pago da derecho al acceso solamente una vez al contenido o servicio, si da el derecho al acceso a varios contenidos, si es por un determinado período de tiempo, o si es por una determinada cantidad de datos, etc. Una descripción más detallada acerca de los distintos modelos de pago se puede encontrar en la 0.

En el ejemplo mostrado en la Figura 4.7 podemos identificar este elemento con el identificador *ID7* e indica que el modelo que se sigue sólo da derecho a acceder a un único contenido.

Elemento LoyaltyInformation

El elemento *LoyaltyInformation* (*Información de fidelización*) permite expresar los esquemas de fidelización que son aplicables a la transacción de pago. Para expresar esta información podemos utilizar diferentes lenguajes ya que es independiente del lenguaje. Un ejemplo de lenguaje de fidelización es el propuesto en [109]. Para identificar el modelo de fidelización usamos un atributo denominado *LoyaltyModelType* (*Tipo de Modelo de Fidelización*). En la Figura 4.7 no hemos incluido ningún elemento de este tipo.

Elemento Credential

El elemento *Credential* (*Credencial*) se utiliza para especificar credenciales que podrían ser requeridas como información adicional al pago como, por ejemplo, el uso de certificados X.509 de atributos [61], SAML Assertions [39], etc. Entonces, para identificar el tipo de credencial requerida hemos definido un atributo denominado *CredentialModelType* (*Tipo de Modelo de Credencial*). En la Figura 4.7 no hemos incluido ningún elemento de este tipo.

Elemento PaymentProtocolMessage

El elemento *PaymentProtocolMessage* (*Mensaje de un Protocolo de Pago*) tiene el objetivo de encapsular el mensaje de cualquier protocolo de pagos. Este mensaje podría estar especificado tanto en XML como en Base64. Este elemento no se utilizará para describir información de pago sino para, posteriormente, intercambiarla.

Elemento PaymentReceipt

El elemento *PaymentReceipt* (*Recibo de Pago*) ha sido definido para solicitar y enviar los posibles recibos de pago que el cliente y el vendedor pueden intercambiar. Se podría utilizar con varios fines:

- Para solicitar un recibo de pago de un determinado tipo.
- Una vez que se realizado una transacción de pago entre el cliente y el vendedor, como justificante o prueba de la transacción realizada.
- Si el pago tiene lugar entre el cliente y el PSP, se utilizaría como recibo del pago y serviría para dar acceso al contenido solicitado.

Conjuntos de elementos y referencias

Por cada uno de los elementos definidos anteriormente, hemos definido un nuevo elemento XML que representa un conjunto de esos elementos individuales. Por ejemplo, un conjunto de instancias del elemento *PaymentProtocol* se denomina *PaymentProtocol**s*** (*Protocolo**s** de Pago*). El resto de los elementos se nombran respectivamente como: *Brand**s***, *PaymentServiceProvider**s***, *Price**s***, *PaymentModel**s***, *LoyaltyInformation**s***, *Credential**s*** y *PaymentReceipt**s***. En el ejemplo de la Figura 4.7 podemos encontrar varios de estos elementos.

Elemento *PaymentDescription*

El elemento *PaymentDescription* (*Descripción del Pago*) se utiliza para unir los conjuntos de elementos y referencias definidos en la anterior sección. Este elemento representa una descripción de pago específica que establece las condiciones para una transacción en particular. Está compuesto de (ver Figura F.1): *PaymentReference* (*Referencia al Pago*), para identificar unívocamente el pago al que referencia, entonces, tenemos un conjunto de elementos que referencian a un conjunto de instancias de los elementos *PaymentProtocol*, *Brand*, *PaymentServiceProvider* y *Price*. También aparecen otros elementos como: *PaymentModel*, para especificar el modelo de pago seguido; *LoyaltyInformationAccepted* (*Información de Fidelización Aceptada*), para indicar los esquemas de fidelización y descuento soportados, *CredentialRequired* (*Credenciales Requeridas*), para indicar el conjunto de credenciales que el cliente debería presentar; *PaymentReceiptsSupported*, para indicar el conjunto de formatos de recibo soportados; *Expiration* (*Fecha de expiración*), para indicar hasta cuando esta oferta es válida; y, *AdditionalInformation*, como mecanismo de extensibilidad para futuros elementos que sean necesarios incorporar a la descripción de la información relacionada con un pago.

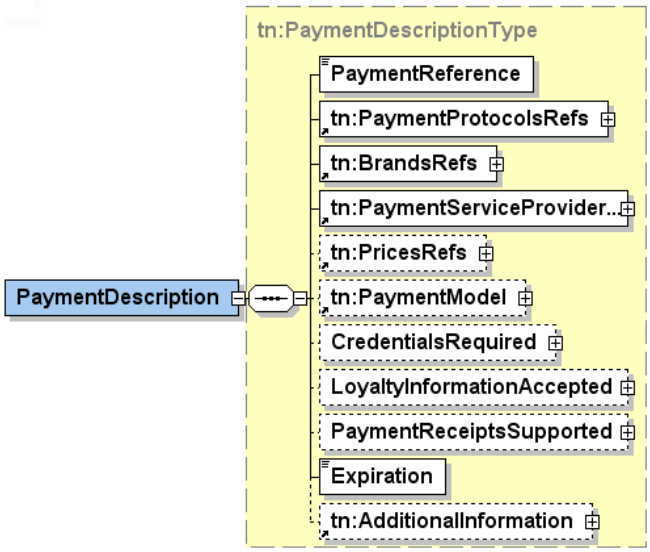


Figura F.1. Elemento *PaymentDescription*.

Este elemento representa los precios a pagar usando un conjunto de protocolos y marcas financieras y, satisfaciendo un conjunto de condiciones tales como presentar un conjunto de credenciales o proporcionar alguna información de fidelización. Entonces, en un elemento de este tipo, cualquier tupla que pueda ser formada por un elemento del conjunto de protocolos y un elemento del conjunto de marcas financieras, representa el mismo precio. Si queremos expresar diferentes condiciones para las diferentes combinaciones, deberíamos crear diferentes descripciones de pago (*PaymentDescription*)

simplificando el número de elementos de cada conjunto. Por tanto, este elemento se utiliza para comparar los precios y condiciones. Además, este elemento define un atributo llamado *negotiable* (*negociable*) que indica si el precio es negociable o no.

En el ejemplo de la Figura 4.7, aparece un elemento de este tipo, con *ID6*, que referencia al protocolo (*SPEED, #ID1*), la marca financiera (*Monedero Euro6000, #ID2*) y el PSP a utilizar para realizar el pago (Research Papers Ltd, #ID3), así como el precio (15 Euros, *#ID4*) y el modelo de pago (un pago un solo uso, *#ID7*).

Elemento *PaymentInformation*

El elemento anterior se utiliza para representar que el servidor está solicitando un pago por el acceso a un producto o servicio. De esta manera contiene los precios solicitados y los mecanismos de pagos soportados. Sin embargo, éste no contiene la información de pago en sí, sólo contiene referencias a dónde está situada esta información. Así, evitamos la repetición de estos elementos.

El lugar para describir tal información de pago es el elemento denominado *PaymentInformation*. Este elemento describe y agrupa toda la información que describe los protocolos, marcas financieras, PSPs, precios, credenciales, información de fidelización, recibos y las diferentes descripciones de pago.

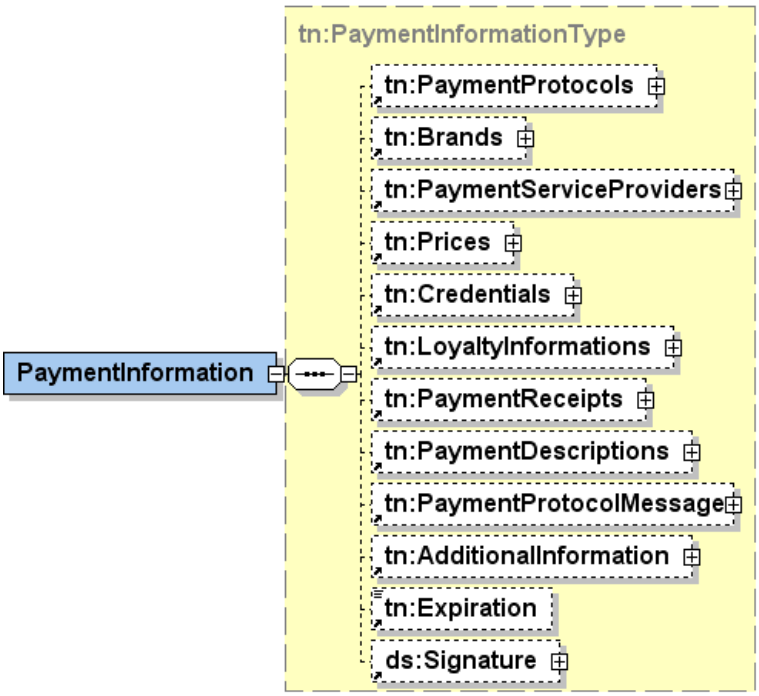


Figura F.2. Elemento *PaymentInformation*.

Este elemento contiene (ver Figura F.2): *PaymentProtocols*, el conjunto de protocolos de pago soportados; *Brands*, el conjunto de marcas financieras soportadas; *PaymentServiceProviders*, el conjunto de PSPs con los que el servidor trabaja; *Prices*, los precios que se usan para expresar las cantidades a pagar; *Credentials*, el conjunto de credenciales que podrían ser solicitadas; *LoyaltyInformations*, el conjunto de los diferentes esquemas de fidelización; *PaymentReceipts*, para intercambiar recibos o solicitarlos; *PaymentDescriptions*, que representa el conjunto de instancias del elemento *PaymentDescription*. Este último elemento introduce los diferentes precios que tienen que ser pagados dependiendo de los protocolos, marcas financieras y PSPs usados. El elemento *PaymentInformation* también contiene otra serie de elementos tales como *PaymentProtocolMessage*, para el caso en que este elemento se utilizara para intercambiar mensajes de pago; *AdditionalInformation*, que sirve como elemento de extensibilidad y que podría ser utilizado para contener futura información de pago que se pueda necesitar; y, un elemento denominado *Expiration* para indicar cuando esta información debería ser actualizada porque deja de ser válida. Opcionalmente, se define un elemento *Signature* (*Firma*) para aquellos casos en los que el servidor quiere garantizar que la información no ha sido modificada. En el ejemplo de la Figura 4.7, como se puede ver, agrupa todos los elementos anteriormente descritos.

Todos estos elementos nos permiten describir la información básica de pago a incluir con los enlaces o cuando se intercambia de información de pago en los distintos frameworks presentados. Todos estos elementos forman parte de un esquema denominado *urn:umu:paymentframework:paymentinformation:v0.1*.

Elemento *EPPMessage*

Estos elementos también se utilizan para incluir la información de pago en los mensajes de EPP que han sido definidos también en XML. Los elementos propios de EPP han sido definidos en un esquema denominado *urn:umu:paymentframework:epp:v0.2*.

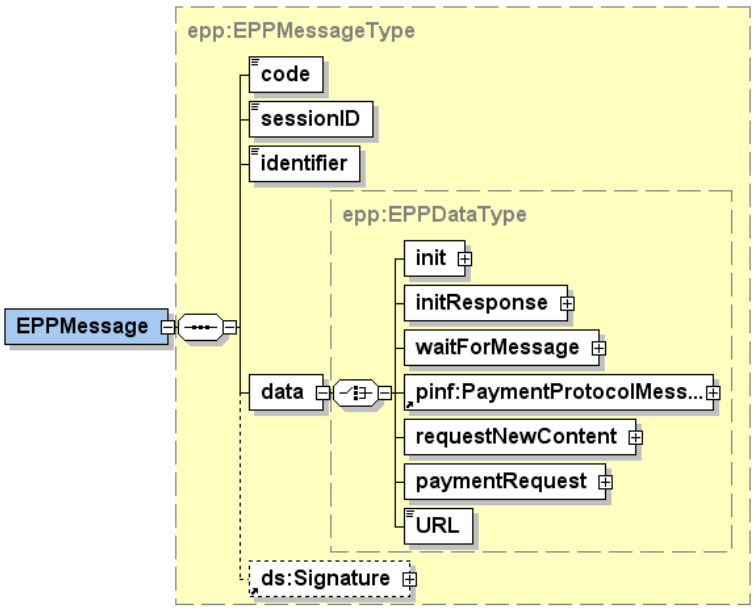


Figura F.3. Elemento *EPPMessage*.

El elemento *EPPMessage* que podemos ver en la Figura F.3 representa un mensaje en EPP. Los distintos elementos que forman parte de estos mensajes no se explicarán ya que su definición se ha llevado a cabo tal y como se describió en la Sección 4.2.3.