Securing group based peer-to-peer systems

Joan Arnedo Moreno

Advisor: Jordi Herrera Joancomartí

Tutor: Joan Carles Cruellas Ibarz

A thesis presented in fulfillment of the requirements for the degree of Doctor en Informàtica

Departament d'Arquitectura de Computadors (DAC)

Universitat Politecnica de Catalunya (UPC)

およぐ時
よるべなきさまの
蛙かな

与謝　蕪村

A frog is swimming.
Always does in a state of
full dedication.
Yosa Buson (1716-1783)

Abstract

Peer-to-peer applications enable a group of users to create a communications framework from scratch without the need of a central service provider. This is achievable via the aggregation of resources each one of them provides, creating a completely distributed collaborative environment based in a flat hierarchy of users, without the need for centralization. Usually, peer-to-peer applications are conceptualized as a global network, without any kind of logical segmentation or segregation as far as resource availability is concerned. At every model, any peer may access any resource available within the network just by being able to reach the peer that provides such resource. Although having a unique huge open network may be desirable for some applications, there are cases in which it might be interesting to create different, but not necessarily disjoint, groups of peers operating under the same global peer-to-peer network.

In order for peer groups to be able to operate effectively in a global peer-to-peer network, additional security services must be provided. These mechanisms should allow peers to be able to prove group membership to other members of the group, so they can be granted access to group resources, as well as ensuring that resource discovery and message exchange between peer group members remain secure. A group may need to limit membership for various reasons, such as ensuring privacy, anonymity or enforcing that peer group members are up to some specific parameter (data shared, performance, computing power, etc.)

The goals of this PhD. thesis are twofold, the reason being the fact that securing a peer group can be divided at two distinct, but interrelated, layers:

- Enabling effective group membership, starting from the process by which any peer becomes part of a peer group and then, following, the mechanisms by which such peer may prove its membership to other group members for the rest of the membership's lifecycle (peer group access control).

- Providing a secure environment for standard operations within a peer group, which functions once any peer's membership to the group has already been established. Typical operations at this layer are those of resource location and retrieval, or messaging.

In order to achieve the former goal, basic group membership and access control scenarios are categorized and formalized as part of the research work in order to assess which are the current challenges. From this study, we present a generic model proposal that fulfils the objectives of autonomy, keeps a pure peer-to-peer model and the possibility to be used in different peer-to-peer frameworks.

The later goal focuses in secure mechanisms in order to provide basic security services to both resource discovery and message exchange. However, in contrast with group membership models, where a generic approach is feasible, peer group operation security is intimately tied to each specific peer-to-peer framework, since each one specifies resource location and messaging primitives in a different manner. For that reason, a specific one has been chosen for the research work: JXTA. Such election is due to the fact that JXTA's architecture is entirely based on the concept of peer groups, since it was the one to first define the concept of peer group, providing an excellent testbed for peer group research.

Agraïments

Si bé aquesta tesi doctoral està escrita íntegrament en anglès, he volgut fer una excepció en aquest aparat. Al tractar-se d'un text on s'expressen sentiments personals, penso que el més addient és fer servir la meva llengua materna, així com la de la majoria de la gent a qui vull expressar els meus agraïments. Espero que el lector comprendrà el meu punt de vista i em sabrà dispensar.

Primer de tot, indiscutiblement, he d'agraïr la tasca realitzada pel meu director de tesi, en Jordi Herrera, sense el qual aquest treball, sincerament, mai hagués estat possible. Especialment per la seva cura a l'hora de revisar la meva recerca i la seva paciència i sentit de l'humor amb el meu peculiar estil d'escriptura en anglès. Juntament amb ell, també vull dedicar unes paraules d'agraïment al meu tutor, en Joan Carles Cruellas, el meu mentor a la UPC des de fa molts anys i principal responsable de les meves inquietuds en el món de la seguretat i l'XML. El seu llegat també es fa evident en aquest treball.

Tot seguit, vull dedicar unes paraules a aquella gent que durant tot el temps que he dedicat a realitzar aquesta tesi han hagut de suportar no haver de suportar-me. En aquesta aparent contradicció, em refereixo a la meva família (inclosos els meus gats) i els meus amics més propers, que tantes vegades han hagut d'escoltar un "Avui no puc, tinc una data límit per un article" o han sofert d'alguna manera les meves abscències per motius diversos. Finalment, aquí està el resultat de la vostra comprensió i suport incondicional. Una part d'aquest resultat també és fruit vostre.

Un altre grup molt important de gent que m'ha donat la suficient embranzida

per arribar fins al final són els meus companys de feina. La nova generació de joves doctors (o que ho seran en breu) de la UOC que han sabut combinar paraules d'ànim i savis consells, fruit de la seva pròpia experiència personal, amb la dosi justa de pressió: "A veure si acabes ja d'una vegada...".

Finalment, no m'agradaria concloure aquesta secció sense donar les gràcies a la gent vinculada al Departament d'Enginyeria de la Informació i la Comunicació de l'Institut de Tecnologia de Fukuoka, i molt especialment en Barolli-sensei, en Keita-san i l'Ikeda-senpai per la seva enorme hospitalitat i gran suport en la meva recta final d'aquest treball durant la meva estada a la seva institució.

Contents

Chapter 1

Introduction

Peer-to-peer (P2P) applications enable a group of users to create a communication framework from scratch without the need of a central service provider. Via the aggregation of the resources each single peer provides, it is possible to create a completely distributed collaborative environment based in a flat hierarchy of users, without the need for centralization. There is no such notion as clients or servers since, ideally, all peers are equal within the network. The goal of such distributed infrastructure is to minimize the system's dependency on very specific nodes, which may produce bottlenecks or single points of failure. The result is a paradigm shift from the basic centralized client-server model.

Despite the initial idea of a fully-distributed environment, P2P applications may operate under three different models, as shown in Figure 1.1. In a centralized model *(a)*, all resources and peer presence services are indexed by a central server, but peers may directly exchange messages in order to share such resources. Actually, the peers' capability to directly communicate without the need of a central server is the only reason why it can be considered a P2P model. In a semi-centralized model *(b)*, a set of peers with additional capabilities, named *superpeers*, exist in order to manage resource discovery and message exchange. Finally, in a pure model *(c)*, all peers are truly equal and must collaborate in order to provide the necessary network services.

An important characteristic of P2P networks is that they operate at the application layer and, for that reason, a basic communication infrastructure at the

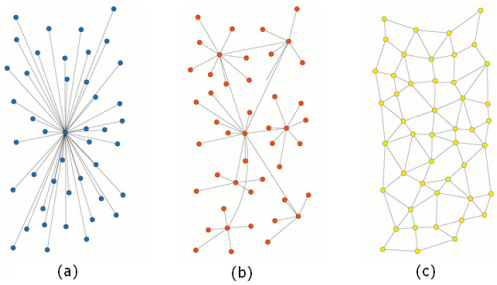


Figure 1.1: *Peer-to-peer models: (a) centralized model, (b) semi-centralized model, (c) pure model*

physical layer is also assumed.

Usually, P2P applications are conceptualized as a global network, without any kind of logical segmentation or segregation as far as resource availability is concerned. At every model, any peer may access any resource available within the network just by being able to reach the peer that provides such resource. Although having a unique huge open network may be desirable for some applications, there are cases in which it might be interesting to create different, but not necessarily disjoint, sets of peers operating under the same global P2P network.

In this scenario, each one of these sets is called a *peer group*. There are several motivations, the most typical ones being:

- *A secure environment.* Peer group boundaries permit members to access and publish protected contents. Peer groups form logical regions whose boundaries limit access to the peer group resources, in a similar way to a VPN [Fer98], where computers may talk to each other across the Internet, but protected from interlopers.

- *A scoping environment.* Peer groups define the search scope for resource look up. Peer groups may be used in order to limit the amount of message exchanges in which a peer takes part. Only those messages deemed interesting by each peer will be processed, limiting traffic to a manageable amount. This is specially interesting under a pure model, where, as previously explained, resource discovery becomes very inefficient as the search scope increases.

- *A monitoring environment.* Peer groups allow peers to monitor a set of peers for any special purpose, including heartbeat, traffic introspection, and accountability.

Under a peer group environment, the network still operates as a whole, providing a common framework, but resource access may be limited depending on group membership. This concept is represented in Figure 1.2: the global overlay network exists, but peers at different locations are members of different peer groups (or of none at all), without the need not be adjacent or sharing the same physical network.

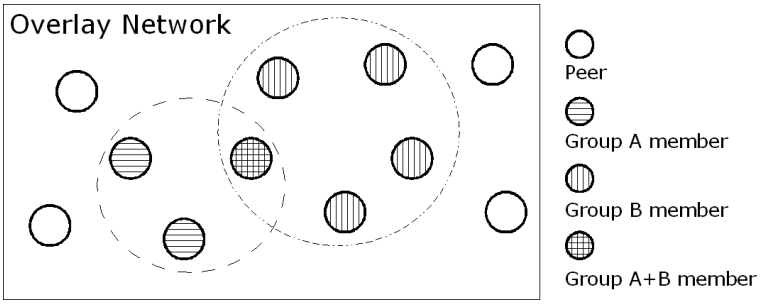


Figure 1.2: *Global overlay network with peer groups*

Just as the popularity of P2P applications has risen, concerns regarding their security have also increased, specially since it is no longer possible to trust a central server which capitalizes all security operations. As P2P applications move from simple data sharing to a broader spectrum, they become more and more sensitive to security threats and it becomes capital to take into account which security mechanisms exist in current P2P platforms before deploying them. Two are the main challenges to the creation of this secure environment:

- Providing the necessary security mechanisms in order to protect transmitted data in an scenario where messages travel through unknown peers (usually, peers which are not group members) across the overlay network.
- Because of the nature of P2P, all these security services should be provided by the peer group members in a decentralized manner and should not rely on external parties.

Security mechanisms are of special importance in a peer group scenario. Some method which allows peers to prove group membership to other group members is necessary, so they can be granted access to the peer group. Furthermore, it should be possible to ensure that resource discovery and message exchange between peer group members remain secure from external interference. Finally, a group may also need to limit membership for various reasons, such as ensuring privacy, anonymity or enforcing that peer group members are up to some specific parameter (shared data, performance, computing power, etc.).

1.1 Research objectives

The main goal of the research work for this PhD. thesis is providing adequate security mechanisms to peer group based environments. To achieve this end, it takes into account the fact that securing a peer group can be divided at two distinct, but interrelated, layers:

- A layer dedicated to group membership, starting from the process by which any peer becomes part of a peer group and then, following, the mechanisms by which such peer may prove its membership to other group members for the rest of the membership's lifecycle (peer group access control).
- A layer dedicated to providing a secure environment for standard operations within a peer group, which functions once any peer's membership to the group has already been established. Typical operations at this layer are resource location and retrieval, or messaging.

Because of the two clear divisions in a peer's group membership lifecycle, this thesis's structure has been divided in two parts: a first one specifically related to securing peer group membership (Part I) and another one related to securing peer group operation (Part II).

Part I defines a group membership and access control method that is adaptable to a wide range of group policies. The approach is based on a fully decentralized infrastructure, a pure P2P model, and pays special attention to its members' autonomy and self-organization. Another key aspect is keeping a pure model even when security mechanisms must be deployed, avoiding at all costs dependency from entities external to the group, which would break the ideary of P2P.

In order to achieve this goal, basic group membership and access control scenarios are categorized and formalized as part of the research work in order to assess which are the current challenges. From this study, a proposal of a generic model that fulfills the objectives and may be used in different P2P frameworks is presented. This proposal takes into account other key aspects such as peer equality.

The main goals of this part can be summarized as:

- Assess how current security approaches in the field of access control apply to the specific scenario of peer groups.
- Categorize the different scenarios for peer group membership and access control according to the involvement and roles of group members and study how current security approaches apply to each of them.
- Define a group membership security model based in a pure model which may be adapted to any of the different identified scenarios.
- Study how some degree of anonymity may be maintained in peer group membership and provide a method which may be applied to the proposed security model.
- Specify the group membership model for a particular P2P system, fully realizing its basic capabilities and architecture to achieve this end.

The chosen system for the last goal has been JXTA [SUN01] (or "juxtapose"), a set of open protocols that enable the creation and deployment of P2P networks promoted by Sun Microsystems since 2001. JXTA is the main contributor to the the concept of a P2P environment where peers operate within the context of a group, being the first system to introduce this concept. Its whole architecture has been designed around this single concept. For that reason, it is the main referent when studying peer group based systems, providing an excellent testbed for research in this field.

Furthermore, as the popularity of JXTA has increased, not only it has become one of the main frameworks for P2P applications (over 2,700,000 downloads, 120+ active projects, 18,000+ members)[SUN01], but different companies already have taken advantage of the provided framework in order to develop products based on a P2P model [Ash04]. Some of them are quite important ones, such as Verizon

or Nokia. The former uses JXTA in a program which directs both voice and data traffic through its network, providing users with an advanced version of instant messaging with telephony tools. The latter uses a JXTA based software system in its data centers as a means to network monitoring and management. Other companies which use JXTA in their products encompass Codefarm [Cod], which uses JXTA to distribute its artificial intelligence algorithms for solving problems such as market financial analysis across many resources, and SNSing, China's most used social networking software.

Part II is concerned in providing a secure environment to peer groups. However, in contrast with group membership models, where a generic approach is feasible, peer group operation security is intimately tied to each specific P2P framework, since each one specifies resource location and messaging primitives in a different manner. For that reason, this part exclusively focuses on JXTA, instead of providing more abstract security mechanisms.

A secure environment for JXTA's peer groups is provided by reviewing its current security mechanisms in order to comprehend its shortcomings and how to improve them. Special care is taken on regards to the typical peer's lifecycle and its basic operations once a peer group has been joined. Once this requirement has been fulfilled, it is possible to specify a secure layer for resource distribution and messaging which takes into account JXTA's unique architecture.

The main goals of this part follow:

- Exhaustively analyze the current state of security in JXTA, providing a comprehensive survey of how its security mechanisms work and its current weaknesses.
- Specify a method for securely distributing and locating resources within a JXTA peer group.
- Extend the current JXTA's core protocols to protect them against typical attacks in P2P networks.

1.2 Document structure

The first part of this work, related to securing peer group membership, encompasses the following chapters:

Chapter 2 provides an in-deep study of the state-of-the-art regarding group membership and access control, outlining the basic methods of authentication and access control to network services. First of all, the classical authentication mechanisms are exposed. Following, the chapter presents a description of the current proposals for ad hoc networks, a field which is extremely similar to the proposed scenario. Closing the chapter, the current trends in P2P systems and how they are related to group membership and access control applications are exposed.

Chapter 3 provides a basic methodology for research, classifying the different scenarios in group membership and access control. Once the classification has been completed, a review of which scenarios may be solved with current proposals is presented, identifying which scenarios are not currently properly fulfilled. A solution for those scenarios is then provided.

Chapter 4 presents a secure and scalable model for group access control and group membership verification in a P2P environment, being the result of the previous scenario formalization. The approach takes into account the nature of P2P networks, being fully decentralized and paying special attention to the autonomy of its members and their self-organization. The proposed model may be applied to any P2P platform.

Chapter 5 defines the specification for the previously defined group membership and access control model using the core JXTA services. The idiosyncrasies of the JXTA platform are taken into account in order to create a system which is fully compliant with its specification. First, the group membership base model is briefly introduced. Then, the group membership specification details are described, explaining how the Membership and Access Services are deployed using the JXTA framework and which are the required support services.

The second part of this work, related to securing peer group operation, is composed by the following chapters:

Chapter 6 provides a survey of the current state of security in JXTA for basic peer operations. Such operations are not analyzed in an isolated way, but the whole peer life cycle is taken into account. The results of this survey provide an up-to-date detailed list of which security vulnerabilities exist and how the current security mechanisms could be enhanced.

Chapter 7 uses the results of the previous chapter's analysis to present a

method based on lightweight key authenticity for securing JXTA's core messaging services: its core protocols and advertisement publication. This method is specifically suited to the idiosyncrasies of JXTA and does not rely on external parties, keeping the P2P model pure, as well as following the ideary of XML standards in order to maximize peer interoperability. Both messaging services are protected against passive and active attacks.

Finally, Chapter 8 concludes this thesis with a summary and discussion of the presented research topics and provides some suggestions for future research.

Part I

Securing peer group membership

Chapter 2

State of the art

This chapter reviews the current state-of-the-art in group membership and access control in distributed environments, such as P2P.

First of all, it is worth mention that, since peers are often dynamic and unknown to each other, it is usually important to be able to provide authentication mechanisms, as noted in [Yun05; NR04]. Nevertheless, peer authentication and group membership are not strictly the same issue. Authentication is related to providing an identity for each peer while group membership is based on access control credentials which may or may not be the peer identity itself.

Although group access control may be achieved without the direct authentication of each node, reviewing authentication models is a basic step in order to solve the proposed scenario, since once it is possible to solve peer identity, which is a harder scenario, then so it is for group membership. Furthermore, being able to authenticate each single peer provides a framework for managing identities in a P2P overlay network. Finally, group access control can be simplified if authentication is solved, since it is always possible to use access control lists as a simple way to filter out which peers are part of the group [NR04].

This chapter is organized as follows. First of all, classical approaches to access control are briefly introduced in Section 2.1, analyzing if they can be applied in a P2P environment. Following, Section 2.2 provides a thorough description of the current state-of-the art in ad hoc networks, a very similar scenario where there's much research done. Finally, in Section 2.3, a review of access control in specific

P2P applications is presented.

2.1 Classical approaches to access control

This section describes the classical approaches that can be applied for authentication and access control. Since such solutions came from the client server-paradigm, they are usually solved in a mainly centralized manner and take management by some administrator for granted.

In all classical authentication mechanisms, access control is managed via a centralized group member list. Once a user is authenticated, the system looks for a match with the user's identity. If a match is found, credentials are granted, and from then on, they may be used to directly access resources without intervention from the central server. Each method is different in the way a user proves its identity and the credentials' format.

The different kinds of credentials used in classical approaches are:

- Username, authenticated using password.
- Cryptographic symmetric keys.
- Certificates generated using asymmetric cryptography.

Most of the proposals reviewed in the following sections fall into one of these three approaches (but modified in order to operate in an infrastructureless environment).

2.1.1 Unix Authentication

In Unix systems, a user authenticates himself by sending a shared secret key which consists of an 8 character password. A list of system users and corresponding passwords is stored in a centralized database (the */etc/passwd* or */etc/shadow* file). In fact, in order to provide better security, the password itself is not directly stored in this file. Instead, the result from applying a one-way hash function to the password and a salt (a random value between 0 and 4095) is used. The standard one-way function used is the DES [FIP77] encryption algorithm with its result expanded according to the salt. In a more general way, any method based on username and password will be referred as the Unix authentication model.

The Unix system also stores in the password file which groups each user belongs to, so once a user is authenticated, the system may automatically retrieve group information. Access control to group specific resources (basically, files) is achieved via permissions individually assigned to each resource.

From a security point of view, Unix authentication is extremely weak. The password is very short and susceptible to brute-forcing, or, at least, dictionary attacks, since it will probably be human readable. In fact, in old implementations of Unix any user could access the password file. Fortunately, in modern systems the password file can only be accessed by processes with super-user rights.

Even though in its initial inception the Unix authentication system was fully local to each system, it is possible to distribute the password file using the Network Information Service (NIS) [Hes92]. However, such solution is still a centralized system with a single point of failure.

Several collaborative environments such as CVS or BSCW base its model of authentication on a username and password approach. Nevertheless, they are also fully centralized systems.

Even though the Unix system can be compared to a peer group scenario, considering that peers would be equivalent to simple users, the Unix system is far from a P2P architecture, since it has infrastructure (the operating system itself). Furthermore, there's no peer equality, since there are super-users with greater privileges which choose which users may join the system and which groups they belong to. It is reviewed only to show the simplest way for basic authentication and group membership (apart from no authentication).

2.1.2 Shared key authentication

Kerberos [Mil88] is one of the most well-known authentication architectures based on shared key, its name inspired from the greek mythology three-headed dog that guarded Hades. Kerberos' goal is enabling network applications to securely identify their peers. In order to prove its identity, clients initiate a three-party protocol with a server and a TTP (Trusted Third Party): the Key Distribution Center (KDC). The KDC consists of two logically separate parts: an Authentication Server (AS) and a Ticket Granting Server (TGS). A Kerberos ticket is only valid for a finite lifetime.

The KDC issues a ticket to the client which will be used to prove client identity

in front of a server and then establish a temporary encryption key between both the client and the server. In order to prevent replay attacks, using the same ticket several times, an *authenticator*, some timestamped data, is also generated during the protocol and presented to the server. After the ticket expires, a new one must be requested from the KDC.

The authentication protocol is shown in Listing 2.1. A and B are respectively the client and the server. S stands for the KDC (both the AS and TGS). K_{AB} is a shared key between A and B. $\{\}_{K_{AB}}$ is some encrypted data by means of a shared key. T denotes a timestamp and L a lifespan:

$$\begin{aligned} A &\rightarrow S : A, B \\ S &\rightarrow A : \{T_S, L, K_{AB}, B, \{T_S, L, K_{AB}, A\}_{K_{BS}}\}_{K_{AS}} \\ A &\rightarrow B : \{T_S, L, K_{AB}, A\}_{K_{BS}}, \{A, T_A\}_{K_{AB}} \\ B &\rightarrow A : \{T_A + 1\}_{K_{AB}} \end{aligned} \tag{2.1}$$

The KDC is trusted to hold secret keys known by each client and server on the network (the secret keys are established via an out-of-band channel). The key shared with the KDC forms the basis upon which a client or server believes the authenticity of the tickets it receives.

Each installation of Kerberos comprises an autonomously administered domain and establishes its own KDC, which only operates within that domain. In our scenario, a Kerberos domain would be equivalent to a group.

Nowadays, Kerberos may be used as means of authentication for applications such as OpenSSH or NFS.

Kerberos was developed as a centralized system in a pure client-server environment (even though some services may be logically separated), not concerned with peer equality. In fact, some services which are critical to the system are localized in a single peer (basically, the KDC), which cannot be guaranteed to be always online in a P2P system and becomes a single point of failure. For that reason, Kerberos may not be directly applied to P2P environments. Nevertheless, a basic idea that will be seen again in many other different approaches is proposed: the use of a TTP as a means to authenticate users (in this case, via shared keys). This approach applies to peer group access control in the sense that Kerberos-protected services can be considered groups, its members being those

peers able to access them.

2.1.3 Certificate based authentication

One of the first certificate based approaches was the european project Sesame [Ash97], taking direct inspiration from Kerberos' architecture and providing some improvements. Its main objectives are the definition and implementation of authentication and access control management. The main Sesame contributions are the use of asymmetric cryptography and attribute certificate based access control. In some way, Sesame was the precursor of the concept of a PKI (Public Key Infrastructure).

In Sesame, each application keeps a short Access Control List (ACL) that mentions the roles that are allowed to access the application. When a user wants to access the application, he forwards his Privilege Attribute Certificate (PAC). The application verifies the digital signature included in that PAC and also verifies that the user who sent it is the same that the one mentioned inside the PAC. Then, the application checks his roles and compares them with its ACL. If the verification is successful, access is granted.

Sesame is composed of three online authorities: Authentication Server (AS), Privilege Attribute Server (PAS) and Key Distribution Server (KDS). The KDS just takes care of shared key generation for each user within a domain, and may be replaced by public key infrastructure if asymmetric cryptography has to be used.

In order to authenticate a user, the protocol in Listing 2.2 is initiated. m stands for a random message or timestamp in order to prevent replay attacks. $Cert(A)$ is a certificate for A. $Sign_A(m)$ is some data m signed by A. TGS is a Ticket Granting Server for Kerberos interoperability.

$$\begin{aligned} A &\rightarrow AS : A, Cert(A), Sign_A(m) \\ AS &\rightarrow A : \{PASsession\}_{K_A}, TGS, Cert(AS), Sign_{AS}(m) \\ A &\rightarrow PAS : PASsession \\ PAS &\rightarrow A : PAC(\text{if access is granted}) \end{aligned} \tag{2.2}$$

As mentioned, Sesame is different from Kerberos since in this model public key certificates are used to enforce access control, instead of exclusively relying on shared keys.

Currently, any application may integrate Sesame authentication via the GSS-API. Unfortunately, the public distribution was made cryptographically weak because of pressures by the European Commission.

Even though Sesame cannot be considered an infrastructureless environment, each service is still in a single node and no peer equality exists, the basic idea remains and might be used in a P2P environment. Just like Kerberos, access to services is equivalent to group access if each service is considered a group.

Another notable proposal for authentication and access control using certificates is Akenti [Mud03]. Apart from identity certificates, based in the X.509 standard [CCI88] with basically the same functionalities as the ones in Sesame, Akenti used two new kinds of certificates: use-condition and attribute ones. The former allows users to specify which conditions must be met in order to allow access to some resource. The latter certifies users as being member of some group. Using these three different types of certificates allows Akenti to obviate the use of ACLs.

In this approach, the identity certificates are not managed by a centralized Certificate Authority (CA). Instead, the system relied on third-party CA certificates, such as those created by Verisign or Entrust. The use-condition and attribute certificates are generated by the system's own users.

Even though shared key approaches to authentication allow access from users in different administrative domains (for example, using cross-realm trust in Kerberos), certificate-based approaches are an improvement since they allow users to easily manage access control via certificates.

2.2 Ad hoc network authentication

The field of wireless ad hoc networking is closely related to group access control regarding the model exposed in Chapter 1. Even though P2P is entirely related to software and ad hoc networking to the physical devices themselves, the basic principle is the same: a group of users are able to create a communications infrastructure from scratch without the need of a central service provider.

In ad hoc networking, one of the main security concerns is creating groups of nodes which may communicate without fear of rogue nodes trying to infiltrate, or even jeopardize, the group itself. We are also under the assumption that any node is part of the global network, since they operate under the same shared medium: the wireless one. This may be extrapolated to a P2P environment in which group access control is needed. In this case, the equivalent to the wireless medium would be the Internet itself.

Nevertheless, it must be never forgotten the fact that in wireless ad hoc networks, the area where a peer may operate is intrinsically limited by its signal coverage range. Two peers who are beyond the scope of their antennas will never be able to directly communicate and must rely on multi-hopping. However, in a P2P application network, no range limitations exist, since the lower network layers guarantee that two peers may directly establish a connection with no initial need for multi-hop. On the downside, no such limitation also means that any peer may always try to infiltrate the group and access its resources. Rogue peers can always reach the group itself. However, in a logical P2P network, rogue peers may not always be able to directly eavesdrop communications between two peers.

[Hoe04] provides a very good review of the different approaches to authentication in ad hoc networking. It must be noted that, even though authentication may be an important step towards group access control, not every authentication system may apply into this context.

Usually, the different approaches to authentication are classified according to which kind of cryptographic system is used: symmetric vs asymmetric. This division only emphasizes issues regarding key management and distribution, such as the need for confidential or authentic channels. Such emphasis is very important, but there are more issues that must be taken into account in our scenario.

This section will be divided according to four different approaches: shared key, pairwise shared key, CA dependant and non-CA dependant approaches. This division makes it much easier to effectively evaluate which are the common features and flaws when trying to apply every different scenario to the P2P environment, since there are very important challenges, in addition to key distribution, which should be seriously taken into account. For instance, availability, scalability, peer equality or self-organization.

2.2.1 Shared key approaches

A shared key approach is the most obvious solution to the proposed scenario of being able to proof group membership in order to be granted access. The same single token (usually a cryptographic key) is shared between all the members within the group. Proof of membership may be achieved via the direct usage of this token.

It must be noted that, in this approach, any node which is member of the group may perform access control. The capability of group access control is not limited to a very specific set of nodes, which may be desirable since it keeps peer equality. In Figure 2.1, any peer which holds the shared key K_1 is able to prove its membership to group A.

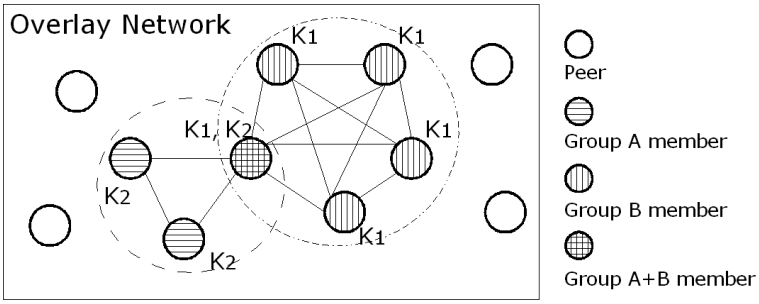


Figure 2.1: *Network with shared key groups*

Key management and distribution are the main issues in this approach. The shared key must be transmitted to new group members via an out-of-band secure channel. Whenever the key must be changed, a new one must be created and transmitted to each group member, which would be equivalent to creating the whole group again from scratch. This may be needed in case of key compromise, but also when a single node has to be removed from the group. Being forced to recreate the group on the event of any member leaving the group is very restrictive and makes this model unsuitable for very dynamic groups or those where its members are not inside the same organization or physically near to each other.

802.11 model

Even though the 802.11 name directly refers to the wireless medium, this model name is just used to encompass any group access approach based on shared symmetric key schemes.

The basic 802.11 security model [IEE99] accepts two authentication models: *open system*, where all nodes may join the group and no access control is enforced, and *shared key*. A cryptographic symmetric key is used as a shared token.

In this model, a challenge-response protocol is enough to prove that any node is in possession of the same key, without having to reveal the key itself or transmitting it over an insecure channel. The shared key may be used for both authentication and encryption.

In the IEEE 802.11 standard, the WEP protocol was originally used for authentication and encryption, but since it was proved that its algorithm implementation was weak [Wal00], it evolved into the 802.11i standard, which uses the AES algorithm.

The WEP weakness comes from the fact that it uses the RC4 stream cipher, and as such, the same traffic key should never be used twice. For that purpose, a Initialization Vector (IV) is used. The IV is a 24-bit field, which is sent in the cleartext part of a message. Unfortunately, it is not long enough to ensure that some key stream will not be repeated on a busy network. Using cryptanalysis, it is possible to exploit the way the RC4 cipher and IV are used, resulting in a passive attack that can recover the RC4 key after eavesdropping the network for a few hours.

In the 802.11i [Che05] standard, discretionary control may be achieved using an authentication server, which distributes different keys to each user. This approach is similar in concept to Kerberos, as shown in Section 2.1.2, and shares the same problems in a pure P2P environment.

Password model

The password model is basically the same as the 802.11 model, but the shared token is a human readable password instead of a cryptographic key. Since passwords are prone to brute force dictionary attacks, this approach is usually reinforced using a password-authenticated key exchange (PAKE) in order to create a strong cryptographic key from the password itself via an asymmetric scheme. An ef-

ficient proposal for this was introduced in [Kat01]. The use of human readable passwords is what makes it different from the 802.11 model, even though the basic concept is very similar.

The initial approach to PAKE was just between two nodes, since it uses a Diffie-Hellman (DH) key agreement [Dif76]. This kind of model may be used in group access control via the approach proposed in [Aso00], which solves n-party DH key agreement.

2.2.2 Pairwise shared key approaches

In this approach, a single token is not shared between all members of the group, but different tokens are shared pairwise between different members of the group.

In this model, nodes are not able provide group access control for any another member which is trying to join the group. Only those nodes which share with him a token may authenticate each other. In this particular case, it does not mean that there's no peer equality, but it may have an obvious impact in group availability. Limiting the sets of peers with the capability to authenticate also requires operating in a multi-hop network, since direct connections between any peer may not be possible. Members may only connect via a specific subset of peers.

This approach is graphically shown in Figure 2.2. Peers which share the K_{13} token may mutually authenticate as members of group B. However, there are cases where peers may not authenticate even when they are actually members of the same group. For example, this is the case between peers which only hold tokens K_{12} and K_{14}.

Group availability is a big concern in this approach, since only a small subset of peers (one, in the worst case scenario) may grant access to a connecting peer. In the previous figure, the node holding only K_{22} may only join the group and access the rest of the peers when the peer holding K_{21} and K_{22} has already joined and is available.

Pairwise key pre-distribution model

This model was created in the field of sensor networks. Instead of using a single common key for the whole group, different keys are assigned pairwise between each peer and compromising one sensor does not compromise the whole group.

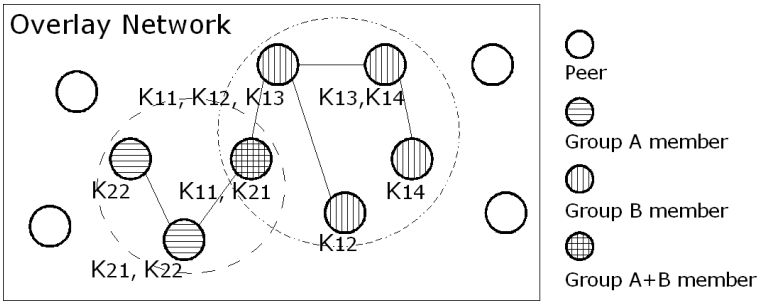


Figure 2.2: *Network with pairwise shared key groups*

However, instead of every peer knowing the key of every other peer in the group (which may be unfeasible in limited memory devices as sensors), only a small subset of keys are stored. The subset each peer stored must be enough to allow all the group to be represented with a connected graph. This set-up process is called pre-distribution.

In [Esc02] a probabilistic pre-distribution protocol is proposed. Each peer is initialized with a subset of keys from a common pool for the group. Membership may be proved if both peers share some key from their subset. If they do not share any key, they try to find a neighboring peer with whom they do share a key and use it as common ground to establish a secure key.

Another approach, much more centered towards sensor networks is presented in [Liu03]. In this proposal the authors take advantage of the assumption that sensors are usually expected to be in static locations and initialize a each one with its expected neighbors' keys. Thus, the probability that two communicating sensors share a key is much higher (or even almost guaranteed).

This model minimizes the impact of key compromise. However, it is assumed that the location and number of nodes is static, which may not be the case in a P2P network. A trusted authority that will take care of pre-distribution is also implied.

A different proposal based on probabilistic pre-distribution may be found in [Tra06]. Here, the specific capabilities of each device within the network are taken into account, introducing the concept of unbalanced random key pre-deployment. Keys are pre-deployed according to an unbalanced distribution, i.e., deploying far more keys in more capable nodes, and fewer keys in less capable ones. This

proposal completely obviates peer equality.

The Resurrecting duckling model

The resurrecting duckling was introduced in [Sta99] and is based on a symmetric key exchange via a secure channel available using physical contact of both devices. Besides the idea of physical contact, the mechanism is basically the same as the previous one.

This model is just enumerated for the sake of completeness, since it only applies when peers are physical devices and does not apply from an application standpoint.

Bluetooth model

The Bluetooth model is the IEEE 802.15 standard for WPAN (Wireless Personal Area Networks) in [IEE99]. In this model, the user must manually enter the password, PIN or key.

Bluetooth registration is normally done with PIN codes. Creating a link between two bluetooth devices is named *pairing* and the user is required to enter the same PIN code on both devices to complete this process. A 128 bit link key will be generated from this PIN like in the password model. From then on, both devices will be able to authenticate each other using his key via an E_0 stream cipher. If the link key is lost, pairing is also lost between the devices .

The main difference of this model from previous ones is that the PIN is manually entered by the user himself on each device.

2.2.3 CA-dependant approaches

Key management is a big concern in group access control. In a shared key or pairwise shared key environment, the key must be distributed to each peer via a secure channel. If the key is compromised, the steps the group must take will be basically the same as recreating the group from scratch. As group size increases, so does the probability of key compromise. Furthermore, dismissing a peer from the group also means that the key must be changed. However, in a pairwise shared key approach, the steps to be taken in case of key compromise are much simpler. New key distribution is limited to only two peers, those which shared

the compromised key. The rest of the group continues to operate normally and is not affected by the key update.

An alternative to shared keys is asymmetric cryptography, providing each peer with some information which is not shared with other members of the group (K_r, a private key), but that will be enough to prove group membership. This is shown in Figure 2.3. Peers do need to exchange some public data (K_p, a public key) that is linked to its private key, but its distribution does not need a secure channel, since it is considered computationally unfeasible to deduce the private key from the public key.

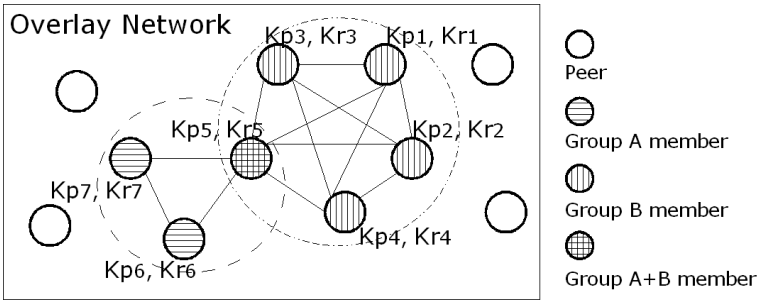


Figure 2.3: *Non-shared key group*

The main concern when asymmetric keys are used is that authenticity must somehow be guaranteed in this exchange since, otherwise, an active attacker may impersonate someone else's public key with his own. Then, the channel must be authentic but not necessarily confidential. CA-based approaches rely on a TTP, implemented as a CA, in order to ensure public key authenticity. The CA will take care of group management operation, acting as an administrator that every member within the group blindly trusts. It will also act as a certificate database that any peer may query.

Centralized CA model

The simplest way to provide CA functionality is assigning such role to a single peer. However, a centralized CA approach is not a good solution in ad hoc networks since it would have a steep impact on availability, as the CA must be online in order for a peer to register to a group or retrieve other peer's certificates.

Relying on a single peer also provides a single point of failure in case of attack. It also goes against the basic principle of peer equality, as only the peer with the CA role may register new members.

In order to improve robustness and availability, the CA may be replicated on n different peers, so it may withstand $n-1$ failures. However, this approach highly increases the system vulnerability to attacks as it also provides n points of failure, so it is not always a feasible solution.

Distributed CA model

Most proposals tend towards equally distributing the CA between a specific subset of peers, but without complete replication of the CA secret. Each peer only knows a part of the secret and they must collaborate in order to retrieve all of it.

In this model, in order to reach a compromise between availability an resistance to attacks, the CA is distributed to n different peers using a (t, n)-threshold scheme. Any t peers must join efforts in order to collaboratively act as a single CA, whereas $t-1$ peers cannot act as a CA.

The distributed CA model tries to keep peer equality by eliminating server nodes and distributing the CA to all the members of the group. Any t peers must collaborate to issue, renew or revoke public keys. The values of parameters n and t must be allowed to be changed while the system is running, since they should depend on the size of the group. Furthermore, if n and t are constant, this model is susceptible to the *mobile adversary* threat. This is also important in order to allow the group to grow.

This basic idea was proposed in [Zho99] with infrastructureless networks in mind. The peers where the CA is distributed are called *server nodes*. t peers must be present at group initialization so they can jointly issue certificates to the new members. Furthermore, any peer may retrieve from them authentic copies of the public keys of any other peer in the group. An additional peer, named *combiner*, is the one which adds up all operations from *server nodes* into a final result. Even though threshold schemes try to minimize impact on availability, the existence of these special peers goes against peer equality. Furthermore, the work load on these peers, using the proposed protocol, is really high.

In a later improvement of the aforementioned proposal [Kon01], workload is reduced by letting each peer manage its own copy of the signed public key, instead

of retrieving it from the server nodes.

However, the threshold scheme approach has some issues regarding how public key authenticity is achieved in case a peer wants to push its public key to the system, which is a very important step in group management. The authors solution is physical contact, which may not be feasible in a P2P environment. Basically, an out-of-band method for authentication is needed.

COCA [Zho02] is the first system to integrate a Byzantine quorum system [Lam82] in order to achieve availability. Such system is a collection of subsets of servers, each pair of which intersect in a set containing sufficiently many correct servers to guarantee consistency to the clients. In this proposal, defense against mobile adversaries is achieved using proactive recovery. In addition to tackling problems related to fault-tolerance and security, new proactive recovery protocols were developed in this proposal. However, all of this has some cost in availability, since the protocol is much more complex and efficient group communication must be guaranteed.

In MOCA [Yi03], a different framework based on threshold cryptography is proposed. In this case, the main motivation is providing an efficient certification protocol, the MOCA Certification Protocol. This protocol reduces the amount of overhead from flooding while maintaining an acceptable level of service, introducing the concept of β-unicast, where the client can use multiple unicast connections to replace flooding if the client has sufficient routes to CA peers in its routing cache. β represents the sufficient number of cached routes to use unicast instead of flooding.

Finally, DICTATE [Luo05] approaches the distributed CA model by dividing the CA itself in two different entities: an offline Identification Authority (IA) and an online Revocation Authority (RA). The IA authenticates the initial binding between a public key and its subject entity, and the RA keeps track of the certificates' status. Thanks to this separation, compromising the online authority, which is usually more vulnerable than an offline one, does not enable the adversary to issue certificates to new users. This proposal still uses threshold cryptography in order to delegate CA responsibility to different peers.

Identity-based model

Identity-based systems [Sha84] obviate the need of an authentic channel to exchange public keys by using some common known information as the public key itself. Usually, such information will be the peers' network identifier. In the case of human beings, it could be directly its name, e-mail address, etc. That means the association between a intended peer and its public key is direct and cannot be forged.

This approach does not avoid the need for a CA at the initial stages of group setup (or at any time a new member wants to join), since a TTP is needed in order to generate and distribute the identifiers. The CA is also the one who generates the private key for each member. In this initial step, an authentic and confidential channel is needed in order for the CA to guarantee that it is sending the private and public key to the correct peer and prevent eavesdropping.

Since the CA now becomes a key escrow, its power can be limited by distributing its responsibilities between n peers using threshold schemes (such as in the distributed CA model). In any other approach, at some time the CA has knowledge of the group member's private keys. A proposal in this regards may be found in [Kha03]. However, it does not solve how a member may receive its private key in a secure way. Again, distributing the CA contributes towards maintaining peer equality.

Self-certified public key model

This concept was introduced in [Gir91] and is also based on public key cryptography, but obviates the need for certificates making the peer's identity part of the public/private key pair. Because of that, the public key itself provides authenticity. In order to achieve this end, the CA generates a self-certified public key for each peer using its own private key and the peer's public key as input. Nevertheless, the CA does not know the private key from each peer. Again, a TTP is needed in order for a group to accept new members and peers must have access to the CA's public key in order to verify self-certified keys.

In the mentioned proposal, an authentication protocol is presented, as well as a DH key agreement protocol, which is resistant to man-in-the-middle attacks because of the self-certified nature of the public keys. However, this protocol yields the same shared key between two parties, given the same base public keys. That

means that a different self-certified key must be generated by the CA every time two parties have to authenticate each other, which is not feasible in a dynamic environment.

It must be noted that this approach is different from the identity-based model since the identity itself is not the public key, but merely embedded inside it.

Key chain model

This model uses hash chains [Lam81] in order to authenticate group members. In this method, a hash function h is applied n times to a seed value x_0. The seed value will be the private key and the final value, $x_n = h^n(x_0)$, will be the public one. In order to prove group membership, a peer is challenged with an x_i value of the hash chain. The challenged peer must respond with the x_{i-1} value, which only the peer who knows the seed value may compute. The computational cost of this operation is very low, since it is based on hashes. However, the calculated hash chain verification may be performed only n times, since from then on it is subject to replay attacks.

Using this approach, there is no need for a TTP or naming infrastructure for identity management. However, an authentic channel is needed in order to exchange public values, and this approach, in its initial form, guarantees peer identity but not peer group membership. The CA signature on the public key is what will provide proof of membership. In fact, in [Wei03], a CA is used in order to provide authenticity. In this proposal, each key chain value ($x_0 ... x_n$) is signed by the CA at different time intervals, allowing peers to verify its authenticity.

2.2.4 Non CA-dependant approaches

There are other approaches which use asymmetric cryptography but do not rely on a TTP for group management. Every member is responsible for generating and managing their own public/private key pair and some other method is used in order to guarantee key authenticity. These approaches are the closest ones to autonomous systems with complete peer equality, but, usually, they need that, some participants trust each other a priori.

Crypto-Based IDentifiers (CBID)

The concept of CBIDs, or statistically unique and cryptographically verifiable IDs (SUCV IDs), was initially conceived for IPv6 addressing in order to solve the issue of address ownership, avoid router supplantation attacks and binding update packet spoofing [Aur; Bon07]. Using this mechanism, each address is automatically bound to a specific node.

Under a CBID scenario, each node generates a public/private key pair. In a P2P scenario, each peer is assigned a unique identifier. Some method that binds the key pair to such identifier is necessary in order to provide authenticity. In the absence of a TTP, this binding is created by applying a pseudo-random function to the public key. The result, or part of it, is henceforth used as the peer address. Any message sent by a peer is then signed using its private key.

In order to validate CBID ownership, some signed message is sent to the authenticating party, usually as part of a challenge-response protocol [cha96]. If signature validation is correct, its is proved that the source peer holds the associated private key. Then the validating public key is used to generate the source peer identifier, *i.e.* the CBID. If the obtained identifier is the same as the claimed one, the key is authentic. By using this method, ID ownership, and therefore authenticity, is guaranteed in a secure manner. However, this this approach does not provide a method for group access control, just identity authenticity.

Certificateless public key model

In this model, a public/private key scheme is used, but no certificates or identifiers are needed. Peers will directly exchange public keys, which will act as their identifiers. The main advantage is that it obviates the need for a naming infrastructure, such as a PKI, which makes things much simpler.

However, such exchange must be done over an authentic channel, since otherwise it is impossible to be sure that the received public key is really the expected one, and not from an intruder. In ad hoc networks, this authentic channel is usually achieved via physical or visual contact between the exchanging devices.

A protocol for public key exchange may be found in [Bal02], which ensures authenticity via *location-limited channels*, which where introduced in the resurrecting duckling model of interaction, but now using asymmetric cryptography.

Again, this approach provides a mechanism to guarantee individual peer iden-

tity, but not group access control. Since there's no CA, it cannot be used as done in the key chain model 2.2.3. Additional measures must be taken in order to register whether a specific peer belongs to a group or not.

Trusted subgroup model

The trusted subgroup model is defined in [Gok03], where the concept of peer group is referred as a *troupe*. In fact, this model is originally proposed for P2P networks, not specifically physical ad hoc networks. It must be taken into account that the proposal goes beyond group membership and tries to provide a distributed trust relationship between members within different groups. As such, only the part relating to *troupe* membership applies to this review.

Members within the same group (or *troup*) collaboratively calculate an RSA accumulator [Nik97]. The accumulator is used as group identifier and is considered the public key. The exponent used by each peer to create the accumulator becomes the private key. Group membership may be proved via a zero-knowledge protocol for modular exponentiation [Cam99]. In this proposal, different operations for group joining and dismissal are provided. However, these operations need heavy computation. In addition, a new key is generated every time a new user joins the group, and as a result, every member must be effectively online when a new peer joins the group.

This model also needs a special unique peer which acts a coordinator during group member join or exclusion operations, the *troup controller*. It must be noted, however, that a troup controller is not the same as a CA, since it does not provide authenticity for public keys. It is simply a coordinator for collaborative operations and any peer may become a group controller.

Self-organized model

In the self-organized model, peers have a public/private key pair and may generate and distribute their own certificates with no need for a TTP. Peers sign certificates for those other peers they trust. This behaves like the PGP web-of-trust [Gar94], and tries to take advantage of the small world phenomenon in order to encompass large groups of peers.

A protocol using this model was introduced an later expanded in [Cap03; Hub01]. In this proposal, every peer has a list of all the certificates he has signed,

peers he trusts, and all from those of peers who have created certificates for him, peers who trust him. When two peers want to authenticate each other, they exchange both lists and try to find a trusted path by merging them.

This model truly keeps peer equality and stays within the spirit of an infrastructureless P2P network. However, it needs out-of-band knowledge in order to prove the authenticity of public keys to be signed by each party.

2.3 Current P2P applications

Studying the field of ad hoc networking, a complete review of the different approaches to authentication in a self-organizing infrastructureless environment has been provided. In this section, group access control for specific P2P applications that work on a logical (not physical) topology will be reviewed.

Roughly speaking, P2P applications are those in which it is enough that each peer may directly communicate with any other one in order to achieve their goals (instead of relaying all messages via a central server). That means that some applications are considered P2P even though they heavily rely on centralized servers or special peers for specific tasks.

Few of the reviewed applications use a truly pure model, maintaining 100% peer equality, the main reason being the quest to guarantee service availability in an easy manner. Depending on which model an application uses, group access control will be easier or much more difficult. Obviously, the most challenging scenario is the pure model.

Even though there is extensive theory behind every listed P2P application, this review will only be focused in those aspects regarding group membership and access control.

2.3.1 Common desktop applications

Filesharing

Most of the current P2P applications are targeted towards file-sharing. In fact, a file sharing application was the one that made the concept of P2P well known across the world: Napster [nap99]. The creation of Napster meant a complete revolution in the Internet and the proliferation of P2P applications. Napster was

shut down by legal action in 2002 because it enabled copyright infringement. This was possible because it worked under the index server model.

Naptser opened the door to new P2P protocols such as Gnutella [gnu00], FastTrack [fas03] or eDonkey [Met02]. Many clients have been developed for these networks (LimeWire, Morpheus, Bearshare, iMesh, Kazaa, eMule, etc.). In fact, some of them, such as Shareazaa, provide an interface for accessing several networks at once. Gnutella is based in a pure P2P model whereas FastTrack and eDonkey have indexing services. Lately, a new protocol gained great renown: Bittorrent [Coh03], which also operates under the index server.

Since the main goal of file-sharing services is to reach the maximum number of users, creating a network as big as possible, none of these applications have group membership capabilities. The whole network is considered the single and only group. Even though some of these applications allow users to provide some *nick* or username, there is no authentication or concept of a unique global identity either. The nearest concept to group membership is that of peer *swarm* in bittorrent, a grouping of peers interested in downloading the same file, but no access control is enforced anyway.

The only file-sharing application that allows some kind of user authentication is DC++ [dc+99]. DC++ peers may connect to different *hubs*, which are basically equivalent to an index server. The list of valid hubs must be obtained via out-of-band methods (usually, public webpages). Hubs may have public access or need user registration. In order to access the latter kind of hub, the hub administrator must provide the new peer with a username and password. It must be noted that in most cases this process is automatized by the DC++ client and server applications and no human intervention is necessary. By default, anybody who asks to register is automatically granted access and only misbehaving peers are expelled from the hub. In that sense, DC++ hub access control is equivalent to Unix authentication as discussed in Section 2.1.1.

As a final note, even though none of the most representative desktop applications provide mechanisms for group membership or authentication, some proposal [Sax03] has been made in order to provide this kind of services to the GNUtella network.

The lack of group membership capabilities in file-sharing desktop applications

does not mean that no effort has been made in order to provide them with some security services. In fact, after Napster was shut down, most efforts regarding security services in P2P applications were directed towards data privacy, anonymity and censorship resistance. In order to achieve these goals, applications such as FreeNet [Cla02], Entorpy [ent], GNUnet [gnu01], Free Haven [Din01] or Turtle [Pop04] appeared. These kind of applications are also referred as the third generation of P2P networks.

FreeNet aims to provide electronic freedom of speech by pooling the contributed bandwidth and storage space of member peers, allowing users to anonymously publish or retrieve various kinds of information. FreeNet uses a kind of key-based routing similar to a distributed hash table to locate peers' data.

Entropy is also designed to be resistant to censorship, much like Freenet. Entropy is an anonymous data store written in the C programming language. The term "Entropy" is an acronym for "Emerging Network To Reduce Orwellian Potency Yield," referring to George Orwell's famous novel Nineteen Eighty-Four and its totalitarian thought-police enslaving people by controlling their information. It was designed to be compatible with the similar FreeNet system.

GNUnet is an official part of the GNU operating system and is based on an anonymity protocol named GAP (GNUnet's Anonymity Protocol) for routing queries and replies. Forwarded query messages are used to search for content and blocks of data. Depending on the load on the forwarding node, messages are forwarded to zero or more nodes. When a node is under stress it drops requests from its neighbour nodes having lower internal trust value.

The Free Haven Project aims to deploy a system for distributed, anonymous, persistent data storage which is robust against attempts by powerful adversaries to find and destroy any stored data. Free Haven hosts the Tor "Onion Routing" software which can make SSL transactions, such as web browsing, anonymous. The Free Haven project was stopped and under study because of detected inefficiencies in the retrieval and reputation systems.

The Turtle architecture is inspired by the way people living in oppressive regimes share information deemed hostile by their government. The main innovation in Turtle is that it builds its overlay network from pre-existing trust relationships among users. Peers only directly connect to a specific set of trusted peers and use indirect routing for queries and results, which provides an acceptable level of protection to each an every peer in the request/retrieval path.

The quest for anonymity also introduced the concept of *friend-to-friend* (F2F) network [Bri00]. A F2F is a particular type of anonymous P2P in which people only use direct connections with their "friends". F2F software only allows people you trust (using IP addresses or digital signatures you trust) to exchange files directly with your computer. Then your friends' own friends, and so on, can indirectly exchange files with your computer via multi-hop. In this way, only peers you trust will know your IP.

This kind of networks are equivalent to pairwise shared key approaches in ad hoc networking, as explained in Section 2.2.2.

The search for anonymity and privacy are different problems from authentication and group membership. However, it must be noted that the quest for anonymity in P2P networks opens new interesting challenges in group access control scenarios, since in this context you cannot base access upon peer identity authentication any longer. The concept of group will be based on some public information, but it cannot be peer identity (for example, peer capabilities, trust, reputation etc.).

Instant Messaging

Some current instant messaging applications also use the described P2P approach in order to allow users to directly communicate in real-time, providing an alternative to server based ones. Applications such as MSN Messenger [Mic05], AOL Instant Messenger [AOL95], Google Talk [Goo05] or Skype [Sky02], which have a high degree of penetration in current desktop environments, use this method.

The most common model for these P2P networks is a simple discovery server, again in order to ensure service availability to the clients. A central server (or group of servers) which are always online, provides presence and location services to clients and manages authentication. Instant messaging applications are based in Unix-like username and password where users may automatically create new accounts via some public web form, with no other human intervention. The only check which is enforced is the fact that no two different users may have the same username and, sometimes, that the new user provides an active e-mail address.

In current instant messaging applications peers may add new contacts they can chat with whenever both parties are online. However, there is no concept such as implicit peer group membership. Peer relationship is absolutely one-to-

one. Again, the fact that B and C are contacts for A does not mean that B and C may initiate any kind of dialog.

Collaboration

Groove [Gro04] is a P2P collaborative application that was designed to enable different workgroups inside a single organization.

Even though Groove is a P2P application, its global security architecture relies on a centralized CA model, based on X.509 standard certificates. Groove provides its own application specific PKI, using management servers. However, since it was conceived with the enterprise environment in mind, it may be integrated with an already deployed external PKI. Groove takes care of all validation procedures in order to ensure that a certificate is valid. Any certificate used within the system must be generated using this centralized CA, which will guarantee each user's identity.

Groove groups are named *workspaces*. Each member keeps a list of current group members' identities, so when a user want to access a resource stored somewhere, its identity may be checked. When a user creates a new workspace, he is the only member. In order for other member to join the workspace, they must receive an invitation from a current member, which is sent via regular e-mail. Once the invitation is confirmed, the new member's identity and cryptographic key information is sent to each group member, so they may authenticate and securely communicate with that user.

Groove's goals are not concerned with peer equality. There are three different kinds of group members: Manager, Participant and Guest. Each group resource may have different permissions with respect to each role, in a Unix-like way. The right to invite new members may also be limited to specific roles.

It must be noted that Groove's security model is a bit different from those described before, since each user within the system will only have one global certificate, no matter how many groups he is member of. A central entity manages the certificates of all users, instead of delegating this task to each individual group. That means that certificate signature only ensures user identity, but provides no additional help for testing group membership. This is not really a distributed approach and is much more similar to an index server model in some way.

Furthermore, as explained in Section 2.2.3, a CA model needs an authentic channel. Groove provides no method for this and relies on out-of-band exchange,

usually phone or personal contact. However, that is not unfeasible in a corporate environment based on a centralized CA.

2.3.2 Peer-to-peer group based middlewares

At this point, well-known P2P desktop applications have been reviewed. However, there's strong research in creating middleware libraries which may enable developers to create their own P2P applications. These libraries do not provide any network client, they only provide a basic abstraction layer for distributed service directives and protocols using P2P, so any kind of client may be created.

JXTA

Reviewing JXTA [SUN01] extensively is a very important step for this work, as it was the specification that introduced the concept of peer group in an overlay network. As a result, it is a point of reference when discussing peer group membership in logical networks. At his stage, it is sufficient to present just a general outline of which mechanisms exist in JXTA to achieve peer group membership and access control and how they are related to the previously described classical approaches. A more thorough review will be exposed in Chapter 5. In addition, a complete security analysis will be provided in Chapter 6.

JXTA is defined as a set of XML based protocols that allow any device connected to a network to exchange messages and collaborate in spite of the network topology. Its goal was to enable a wide range of distributed computing applications by developing a common set of general purpose P2P protocols with absolute platform independence, overcoming many of the limitations found in many P2P applications.

Peers self-organize into *peer groups*: a collection of peers that have a common set of interests. However, the JXTA protocols do not dictate when, where, or why peer groups are created, only describing how peers may publish, discover, join, and monitor peer groups. Peer group membership is managed by two core services: the Membership Service and the Access Service. By default, all peers always belong to a hardcoded peer group named *netPeerGroup*, which would be equivalent to the whole overlay network itself.

When a peer wants to join a specific peer group, it must apply to the Membership Service. The Membership Service manages identities within a peer group,

providing each group member a *credential*. Whenever a resource is accessed within the context of a peer group, the credential is presented. The Access Service then checks its validity, verifying that the sender is entitled to access the resource.

The credential is an opaque token that can be included within each message sent across the network. The source address placed in the message envelope is cross-checked with the sender's identity in the credential. The final exact form of the credential is not set in the specification, each credential implementation is specified as a plug-in configuration, which allows multiple authentication configurations to co-exist on the same network.

The structure of JXTA messages enables applications to add arbitrary metadata information to messages, such as credentials, digests, certificates, and public keys. Message digests guarantee the data integrity of messages. Messages may also be encrypted and signed for confidentiality and non-repudiation. Credentials can be used to provide message authentication and authorization.

This means that JXTA just provides a basic framework and set of protocols, leaving specific implementations and policies up to the developer and providing no specific guideline about how to achieve data security. Initially, any kind of authentication may be implemented under JXTA. However, the current release of JXTA provides two premade specifications for the Membership Service.

The simplest one is the *Passwd Membership Service*, which provides a Membership Service implementation based on a password scheme similar to the Unix system as described in Setcion 2.1.1. This implementation, however, is intended just as an example and is not considered secure.

A more secure implementation is provided using the *PSE* (Personal Security Environment) Membership Service. This service may be used for PKI functionalities in order to provide secure identities [Alt03]. However, as presented, PSE is more of a kind of toolbox that allows the implementation of different models based on securing identities via digital certificates. It provides no clear structure of how trust is managed in a peer group (whose signatures are trusted or which peers are allowed to sign certificates).

The original creators of JXTA do propose a specific membership model using PSE in [Yea03] as a general guideline on how to use it. Some specific, well protected, peers are given CA status for each peer group. Group member candidates must be authorized via an LDAP directory in order to request a certificate and root certificates are distributed out-of-band by being directly included into the

JXTA libraries. This mechanism is akin to a replicated CA approach as explained in Section 2.2.3. The proposal does not specify who configures or manages the LDAP service. It is assumed that some group administrator will do it, which may make it a logical approach in an enterprise environment, but completely contradicts the pure P2P model.

On regards to the Access Service, the current JXTA reference implementation also offers two secure specifications. The *simpleACL* Access Service uses ACLs in order to establish which identities may perform the different group operations. The *PSE* Access Service uses PKIX certificate path validation in order to decide whether the operation is permitted or not. However, currently, none of them really checks group membership, they are just concerned with controlling which identities may perform which operations within the group context.

Other peer group based middlewares

Apart from JXTA, there are some other proposals for middlewares designed towards the creation of peer groups and collaborative applications.

LaCOLLA [Mar03] pays special attention to the autonomy of its members and their self-organization. Its architecture is organized in five kinds of components which behave autonomously. Users may decide to instantiate any number of the following components in the peer they are using:

User Agent (UA): Interacts with applications. Through this interaction, it represents users (group peers) in LaCOLLA.

Repository Agent (RA): Stores objects and events generated inside the group in a persistent manner.

Group Administration and Presence Agent (GAPA): In charge of the administration and management of information about groups and their peers. It is also in charge of peer authentication. Whenever a new peer is authenticated, the GAPA will take care of messaging in order for other members to know that a new peer has joined and is online. Without at least one instantiated GAPA, no peer can be authenticated.

Task Dispatcher Agent (TDA): Distributes tasks to executors. In case all executors are busy, the TDAs queues tasks. Also guarantees that tasks will be executed even though the UA and the member disconnects.

Executor Agent (EA): Executes tasks.

In order to access the components, LaCOLLA provides an API that can be easily used by any application to create collaborative environments. Its API is divided in two parts. One part provides an interface to client applications via a *User Agent* (UA). The other part is used by the UA to notify events or information coming from applications connected to the UA.

LaCOLLA's group access control is based on a Unix model and is managed by GAPAs. The list of group members is replicated in a distributed manner between the different GAPAs inside the group. No different roles are enforced within the group, all users are equal and may invite new members to the group without restriction.

Janus [Sun03] provides another framework where self-organizing peers aggregate in a controlled manner and use communications primitives to accomplish collective goals. This model is implemented in the form of a library and toolkit which provide a family of composable message propagation primitives with strong semantics that enable group communication.

In the Janus architecture, each peer holds a *template* that defines group-specific capabilities. When a peer is initialized, it tries to discover other peers with a corresponding template using any group dependant mean, as well as established methods such as local-subnet broadcasting, IP multicast-address broadcast or network crawling. If no template match is found, the peer temporarily operates as a single node group, but continues scanning the network.

Groups are collections of peers with matching templates that form a connected graph. These definitions allow diverse scenarios such as single node-groups, multiple disjoint peer groups or groups based purely on function. This architecture does not propose how to securely provide access control methods. Any peer with a matching template will automatically be accepted into the group.

Chapter 3

Group membership and access control scenarios

This chapter presents a study [AM06a; AM07] of peer group membership and access control as a first step towards providing an acceptable solution to group membership: the process by which a peer becomes part of a peer group and is able prove at any stage such membership to other group members.

In order to study the security issues related with group membership and access control in P2P applications, the set of possible *scenarios* are identified. We define an scenario as the set of conditions and peer roles under which group membership and access control take place. The scenarios are categorized and formalized as part of the research work in order to assess which are the current challenges. Once such classification is completed, a review of which scenarios may be solved with current literature proposals is presented, identifying which scenarios are not currently properly fulfilled. Therefore, a solution for those scenarios is then provided.

The contributions of this chapter are twofold. First of all, it presents a basic methodology that may be applied to evaluate different approaches to group membership and access control. The resulting classification of each scenario thus allows to identify which scenario fulfills some system requirements and which are the literature proposals that apply towards its solution. On the other hand, the exhaustive classification of the different possible scenarios has allowed to identify

some situations where the literature proposals do not completely or efficiently solve group membership and access control procedures.

This chapter is organized as follows. Section 3.1 presents the different scenarios for group membership and access control in a distributed collaborative environment. In Section 3.2, we discuss the properties and constrains of the defined scenarios. Section 3.3 provides a literature review in order to identify which are the existing solutions for every different scenario. Section 3.4 proposes how to solve the different scenarios under a web-of-trust bases approach. Closing this chapter, a brief chapter summary is provided in Section 3.5.

3.1 Group membership scenarios identification

The membership process is divided in two different steps, registration and authentication, that can be defined as follows:

Registration: The process by which a new peer applies to be accepted into the group. During this process, the new peer may receive any credentials (keys, passwords, tokens, etc.) that will be needed at later stages to prove he belongs to the group. It is assumed that registration is performed only once, and, if the process succeeds, the new peer will be considered a member of the group afterwards.

Authentication: The process by which a user connects to a group, proving he is one of its members. The previous registration process provides the needed evidences for the authentication procedure. In this environment, we identify connection as the possibility to share some resources.

In a pure P2P approach, registration and authentication should be performed by peers within the group without assuming any external party. For that reason it will be useful to provide the following notation:

- Let $G = \{A_1, A_2, \cdots, A_n\}$ be the peers of the group G.
- Let B be the new peer who wants to access the group.
- Let $\Gamma^R = \{A_{i_1}, A_{i_2}, \cdots, A_{i_r}; i_j \in \{1, \cdots, n\}; \forall j = 1, \cdots, r\}$ be the registration structure within a group. That is, the set of r peers who are allowed to register a new peer.

- Let $\Gamma_B^R = \{A_{k_1}, A_{k_2}, \cdots, A_{k_p}; k_j \in \{i_1, i_2, \cdots, i_r\}; \forall j = 1, \cdots, p\}$ be the set of p peers who did register B, where it is obvious that $p \leq r$ and $\Gamma_B^R \subseteq \Gamma^R$ holds.

- Let $\Gamma^A = \{A_{l_1}, A_{l_2}, \cdots, A_{l_q}; l_j \in \{1, \cdots, n\} : \forall j = 1, \cdots, q\}$ be the set of q peers who may authenticate B.

- Let $\Gamma_B^A = \{A_{m_1}, A_{m_2}, \cdots, A_{m_t}; m_j \in \{l_1, l_2, \cdots, l_q\}; \forall j = 1, \cdots, t\}$ be the set of t peers who did authenticate B, where it is obvious that $t \leq q$ and $\Gamma_B^A \subseteq \Gamma^A$ holds.

Both the registration and authentication steps present several common issues and challenges that must be taken into account in order to evaluate the different scenarios:

- Peer equality. Avoiding that some peer has more authoritative power than some other ones in the same group. In an ideal P2P environment, equality should be maximized.

- Availability. Providing an adequate response to a highly dynamic environment, where the number of peers connected to the group in an specific instant may change very quickly over time. Under some circumstances, there may not be enough available peers in order to let B enter the group.

- Acceptance policies. Enforcing some policy in order to decide whether a peer is accepted into the group. This issue is specially interesting in the registration scenario, since we start with no previous knowledge from B (in an authentication scenario we have some knowledge acquired during registration) and such policy must be shared and followed between all members of the group.

- Collusion. Taking into account that under the case that some peers are compromised, it may be possible to leverage the acceptance policy in order to let any peer enter the group.

Since the review of the different scenarios is determined by the involvement of peers, two cases will be differentiated: **single-peer** and **collaborative** scenarios. The former assumes that only one peer is needed in the process, that is $p = 1$ for registration and $t = 1$ for authentication. On the other hand, the collaborative

	r = 0	$0 < r < n$	r = n
$\Gamma_B^R = \{A_i; \text{for some } A_i \in \Gamma^R\}$ case $p = 1$	1R	2Rs	3Rs
$\Gamma_B^R = \{A_{k_1}, \cdots, A_{k_p}; k_j \in \{i_1, \cdots, i_r\}; \forall j = 1, \cdots, p\}$ case $1 < p \leq r$	1R	2Rc	3Rc

Table 3.1: *Registration Scenarios*

case considers $p > 1$ and $t > 1$ respectively. Such distinction has been made because the specific properties of each case strongly impact the system and the security tools needed for every solution.

In the next subsection the different scenarios for registration and authentication are described.

3.1.1 Registration scenarios

Based on the single-peer and collaborative distinction defined above, the possible scenarios for the registration process are summarized in Table 3.1, depending on the number of peers, r, that are allowed to register.

Scenario 1R:

This case, where $\Gamma^R = \emptyset$, is a degenerated case since $p \leq r$ and $r = 0$ but $p \geq 1$. Thera are two different interpretations.

The first one considers the group G as a closed group, no registration process exists so no new peers may be registered. Only the initial peers inside the group will ever be part of it. Since there is no available registration process, both a collaborative effort or single-peer registration may be considered the same case.

The second interpretation is the opposite, and assumes that the group is completely open, no registration process is required. By default, everybody is part of the group and according to the previous registration definition, any peer may create its own credentials by himself, since nobody is responsible for registration. It is worth mention that, under this scenario, the concept of group as a segregate set of peers does not really exist, so it does not really make sense in a group access study, but it is listed for the sake of completeness.

Scenario 2Rs:

In this scenario, only a subset of peers may register new ones, but each one of them may do it on its own ($p = 1$).

Scenario 3Rs:

This scenario tries to keep equality between peers, by allowing every user in G to register a new peer without restrictions.

Collaborative scenarios. 2Rc and 3Rc:

In both cases, a minimum number of peers $p > 1$ must agree before registering B. Such strategy tries to minimize misbehavior from single malign peers (or compromised ones), since they are not able to register B alone.

The main difference between cases 2Rc and 3Rc is the fact that the latter does not make any distinction between peers, since all of them may register B ($r = n$). Furthermore, a limit situation is achieved when $p = r$, meaning that all peers in Γ^R must agree when registering new members.

3.1.2 Authentication scenarios

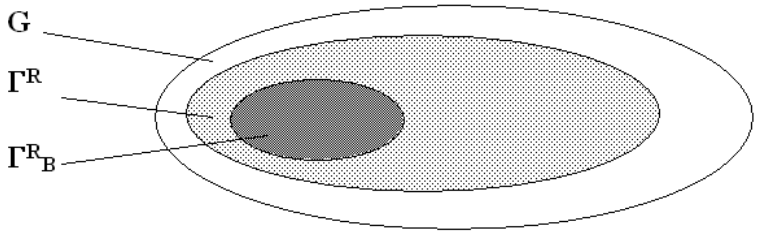
The authentication process allows B to prove his group membership and it will be performed after the registration process.

The authentication scenarios are divided according to the same approach used in the registration process, single-peer cases and collaborative cases. However, since the authentication process is based on a previous registration procedure, the scenarios here are based on a three tiered approach (see Figure 3.1). Now we may take into account the set of peers responsible for registering B, Γ_B^R, when defining the set of authenticating peers.

Then, for the authentication process, the possible scenarios are the ones described in Table 3.2, depending on the number of peers, t, that are allowed to authenticate.

Scenario 1A:

In this scenario there is no authentication process. That implies the registration process becomes completely useless and only makes sense under the 1R scenario.



Despite the exact figure drawn, it is worth mention that we are not assuming only the case $\Gamma_B^R \subset \Gamma^R \subset G$ since equality may be possible ($\Gamma_B^R \subseteq \Gamma^R \subseteq G$).

Figure 3.1: *Registration tiers*

	$t = 1$	$t > 1$
$\Gamma^A = \emptyset$	1A	1A
$\Gamma^A \subseteq \{A_1, \cdots, A_n\} = G$	2As	2Ac
$\Gamma^A \subseteq \Gamma^R$	3As	3Ac
$\Gamma^A \subseteq \Gamma_B^R$	4As	4Ac

Table 3.2: *Authentication Scenarios*

Again, this scenario may be interpreted in two completely different ways.

If we assume that the authentication process provides a mechanism for connecting to the group, then the lack of any authentication process means nobody may connect to the group, so the group becomes closed.

On the other hand, this scenario may be regarded as absolutely free access: any peer will be considered part of the group (being able to share resources) by default. Again, the concept of group itself vanishes.

Scenario 2As:

In this scenario any peer may authenticate B by itself, thus providing maximum flexibility to the system but increasing its vulnerability to compromised peers.

Scenario 3As:

In this case, only peers who were allowed to register new peers may authenticate B. This scenario will typically be based on a 2Rs or 2Rc registration case, where

only a subset of peers, Γ^R, may register new ones ($r < n$). Otherwise, $\Gamma^R = G$, so we would be at scenario 2As.

Scenario 4As:

In this scenario, it is considered that only peers who took part in the registration process, Γ_B^R, may authenticate B.

It is interesting to note that this scenario highly depends on the registration case. Increasing the number of peers in Γ_B^R restricts the registration procedure but provides more flexibility to the authentication process since more peers will be available (notice that this is only true for the single-peer case where $t = 1$).

This scenario may be simplified under some circumstances. When $\Gamma_B^R = G$ in the 3Rc case, it becomes a 2As scenario, whereas in the case that $\Gamma_B^R = \Gamma^R$ in the 2Rc case, it becomes a 3As scenario.

Collaborative scenarios. 2Ac, 3Ac, 4Ac:

In all these collaborative authentication scenarios $t > 1$ peers must agree before B is granted access into the group. The description is basically the same as the one for registration scenarios (see subsection 3.1.1).

Mixed scenarios:

Each of the base scenarios previously defined may be combined producing a new subset of scenarios in which peers from explicitly different tiers may authenticate B before granting access to the group[1].

The different possibilities for mixed scenarios are shown in Table 3.3 where the "+" symbol will denote that the produced blended scenario is composed using the base scenarios indicated. For example, under scenario 3Ac+4As, B would be considered authenticated if $t > 1$ peers in Γ^R agree or $t = 1$ in Γ_B^R agrees.

In fact, mixed scenarios are equivalent to giving different levels of trust to the authenticating peers in each tier (similar to assigning weights).

However, these 20 mixed scenarios may be simplified, since some of the base cases are dependant because $\Gamma_B^R \subseteq \Gamma^R \subseteq G$ holds. In order to show this simplification, we will define the following terms for this section:

[1] Obviously, scenario 1A does not apply here.

2As+3As	2Ac+3As	2As+3Ac	2Ac+3Ac
2As+4As	2Ac+4As	2As+4Ac	2Ac+4Ac
3As+4As	3Ac+4As	3As+4Ac	3Ac+4Ac
2As+3As+4As	2As+3As+4Ac	2As+3Ac+4As	2As+3Ac+4Ac
2Ac+3As+4As	2Ac+3As+4Ac	2Ac+3Ac+4As	2Ac+3Ac+4Ac

Table 3.3: *Initial Mixed Scenarios*

2As	2Ac+3As	2As	2Ac+3Ac
2As	2Ac+4As	2As	2Ac+4Ac
3As	3Ac+4As	3As	3Ac+4Ac
2As	2As	2As	2As
2Ac+3As	2Ac+3As	2Ac+3Ac+4As	2Ac+3Ac+4Ac

Table 3.4: *Simplified Mixed Scenarios*

- Let t_1 be the number of peers who must agree in a 2A scenario in order to authenticate B ($t_1 = 1$ in 2As and $t_1 > 1$ in 2Ac).
- Let t_2 be the number of peers who must agree in a 3A scenario in order to authenticate B ($t_2 = 1$ in 3As and $t_2 > 1$ in 3Ac).
- Let t_3 be the number of peers who must agree in a 4A scenario in order to authenticate B ($t_3 = 1$ in 4As and $t_3 > 1$ in 4Ac).

Then, it holds that:

- Whenever $t_1 \leq t_2$, all mixed scenarios which include a 2A and 3A become a 2A scenario, since $\Gamma^R \subseteq G$.
- Whenever $t_1 \leq t_3$, all mixed scenario which include a 2A and 4A become a 2A scenario, since $\Gamma_B^R \subseteq G$.
- Whenever $t_2 \leq t_3$, all mixed scenario which include a 3A and 4A become a 3A scenario, since $\Gamma_B^R \subseteq \Gamma^R$.

Then, Table 3.3 can be simplified into Table 3.4.

After this transformation, we can see that there are actually only 8 mixed scenarios. In fact, some of them can be further simplified for specific values of t_1,

t_2 or t_3, according to the previous rules. For example, a 2Ac+3Ac where $t_1 = 2$ and $t_2 = 4$ may be simplified to a 2Ac scenario (with $t_1 = 2$).

3.2 Scenario properties and constraints

Once the different available scenarios regarding group access control have been defined, this section discusses the different properties of every scenario, as well as their constraints, in order to asses how they may impact the system.

In scenarios 1R and 1A no real access control exists, so they do not deserve much interest. Under a completely open approach there is neither real security nor groups. On the other hand, the opposite scenario, a closed one, is the least dynamic but provides tighter security. Both cases are the easiest to implement, and strictly speaking, keep peer equality.

Single-peer scenarios are the ones which offer maximum flexibility and responsiveness, at the cost of presenting a single point of failure. The corruption of a single peer is enough to compromise the system.

In collaborative scenarios, for both registration and authentication, higher security is achieved since they are resistant to peer misbehavior or compromise. However, a protocol for collaborative agreement may be needed. Such protocol should be based on some kind of policy upon which agreement between peers is achieved. This extra protocol implies some overhead, then reducing responsiveness.

Another constraint in the collaborative environments is the fact that, under some circumstances, connection may become impossible unless a minimum number of peers are connected to the group, creating a sort of chicken-egg problem. There are two different approaches in order to minimize this constraint. The most straightforward one would be using dynamic collaborative parameters (p for registration, t for authentication), allowing changes in its base requirements during group operations. This is specially critical in registration scenarios when the group itself is still establishing. Another possibility would be delegation upon other peers, which would temporally act as proxies for some peers in Γ^R or Γ^A. This problem is specially critical in physical devices' ad hoc networks: in fact, it is not enough that those peers are online, they must be within transmission range. However, in a P2P environment, range is not an issue.

Taking another approach, the different scenarios may be evaluated from a

peer equality point of view. In scenarios where $\Gamma^R = G$ or $\Gamma^A = G$ (3R, 2A respectively) peer equality is preserved. On the other hand, scenarios in which a restricted set of peers may control group access (2R, 3A, 4A, where $\Gamma^R \subset G$ or $\Gamma^A \subset G$ respectively) lead to different degrees of inequality. Such degrees will require group policies in order to determine which peers belong to these restricted sets.

3.3 Current proposals for the discussed scenarios

This section provides a review to identify which security schemes from Chapter 2 provide a solution to the different scenarios. All terms used in this section are those previously defined.

It is important to point out that a scenario is only considered solved if all restrictions can be strictly guaranteed. This is specially true in scenarios where only a particular set of peers may register or authenticate. It must be completely guaranteed that only one specific set has the effective means to grant access control, and no one else.

3.3.1 Registration scenarios

Scenario 2Rs:

In this scenario, only a restricted set of peers may register B. First of all, it is important to point out that shared key and pairwise shared key approaches are unable to directly solve this scenario, since it is impossible to restrict which peers may grant group access control services only via the chosen cryptographic approach. The necessary knowledge to register a new peer, the shared key, is widely available to all members of the group. The same applies to a self-organized model.

The most widely accepted way to solve access control under this scenario is using a CA, which provides public key certificates to the rightful members of the group. These certificates serve as a credential for the authentication procedure and proof of group membership. Even though some of the different proposals described in Section 2.2.3 use different cryptographic knowledge as a private key (asymmetric keys, ids, key chains), the most important factor in order to match

them with the access control scenarios is which kind of CA is used, not the format of the keys itself.

When $r = 1$, scenario 2Rs is solved using a fully centralized CA approach, with a single point of registration. In this scenario, we must completely rely on the peer which provides CA operations. In order to allow B to be registered, the peer providing CA operations must be connected to the group, and this feature may not be desirable in a dynamic P2P environment. When $r > 1$, the CA must be replicated in every peer in Γ^R, so every peer with full CA functionalities may generate a certificate for B.

The main difference between the previous ways to solve this scenario and a CA based approach is that in the latter case it is possible to limit which peers may register new members without the need of group policies. Peers without CA functionalities are completely unable to register new peers since they do not have access to the CA private key.

Scenario 3Rs:

This is the simplest scenario: any member may register B. The most straightforward method to solve this scenario is using a shared key approach. In this case, registration means providing a copy of the shared key to the new peer B, of which all peers hold a copy.

Pairwise shared key approaches are very similar to shared key approaches. Even though there are different keys within the system, it is enough for the new peer B to share a single key with any peer in order to join the group. This means that any single peer may register B, without any restriction, which also fulfills scenario 3Rs.

The same applies for a self-organized model, since it is based on a web-of-trust, which means that any peer may register B. Once B's public key is signed by a current member, it joins the group. In that sense, it is similar to a pairwise shared key approach, any peer may grant access to a new one via key exchange, but using asymmetric cryptography instead of shared keys.

Finally, a CA approach is also able to solve this scenario, by simply replicating the CA to every peer, a generalization of the solution for scenario 2Rs. However, replicating a CA to every member of the group highly increments its vulnerability and turns this approach into a shared key one (where now the shared key between all members is the CA private key). It must be noted that this approach goes

against the philosophy of having a CA, which is centralizing the private key to strong peers which cannot be easily compromised.

Scenario 2Rc:

There are few proposals which try to solve group access control in a collaborative way. For the scenario in which only a restricted set of peers may register new members, a threshold cryptography distributed CA approach is needed in order to fulfill the requirements. For this specific case, the threshold cryptosystem parameters are (p, r): r peers have the capability to register B, and p peers must collaborate in order to do so.

Scenario 3Rc:

In order to solve scenario 3Rc absolutely all members must collaborate to let B join the group.

Generalizing the threshold cryptography CA approach to every peer in the group, the case where $t = n$, is a straightforward proposal that solves this scenario. In fact, this is the typical approach in threshold CA proposals.

Another approach is the use of a trusted subgroup based on RSA accumulators. In order to join a group, B must provide its own seed and accumulator contribution (see Section 2.2.4 for details). This changes the current group identifier (public key). Even though the group controller, a single peer, stores all intermediate values and knowledge necessary to generate the new group identifier from B's contribution, the new identifier must be broadcasted to each of the current group members. That means that, effectively, every member must be online in order to continue to be able to access the group, which would be equivalent to a 3Rc scenario. Otherwise, they will not know the new group identifier and will not be able to access the group (or they will keep using the old identifier). As a result, even though all the operations needed to create the new group identifier are computed by a single peer, all peers must be present in order for the group to continue to operate as a whole.

In Table 3.5 we can summarize of all solved scenarios:

Scenario	Proposal that can solve it
2Rs	Replicate single CA
3Rs	Shared key, Pairwise shared key, Self-organized, Replicate single CA
2Rc	Threshold distributed CA
3Rc	Threshold distributed CA, Trusted Subgroups

Table 3.5: *Solved registration scenarios*

3.3.2 Authentication scenarios

Scenario 2As:

The scenario where any peer may authenticate B is the most typical one, and the approach most proposals are geared to, since it provides high flexibility and dynamism.

Any shared key approach can solve this scenario, since all peers hold a copy of the shared key, which constitutes proof of group membership itself.

Any CA based approach, be it centralized or distributed, also fulfills this scenario. Once the new peer has a certificate, authentication procedures can be completed via a simple challenge-response protocol in order to prove that the new peer holds the associated private key, and then checking that the certificate is correctly signed by the group CA. In this case, the CA signature provides proof of membership.

The trusted subgroup approach also solves this scenario. In this case, the CA signature is replaced by the fact that only peers which contributed to the RSA accumulator creation may reproduce it. Since any peer may do this, all of them may authenticate.

Scenario 4As:

Even though the scenario where only peers in Γ_B^R may authenticate B seems a bit convoluted, in fact, the pairwise shared key proposals are exactly inside this scope: only the peer which registered B may authenticate it. That is because B only established its link to the group via the peer which holds the shared symmetric key. Without that peer, no other one may grant him access, since no key is shared with any other member.

Scenario 4Ac:

Self-organized approaches are based on this scenario. Any peer may try to find a trust chain to B in order to let him connect to the group. If a chain exists, it becomes proof of membership. However, several peers must collaborate in order to retrieve such trust chains.

To our best knowledge, the rest of authentication scenarios are unsolved with the current state of the art proposals.

A summary of all solved authentication scenarios may be found in Table 3.6:

Scenario	Proposal that can solve it
2As	Shared key, CA based, Trusted subgroup
3As	Unsolved
4As	Pairwise shared key
2Ac	Self-organized
3Ac, 4Ac	Unsolved
Mixed	Unsolved

Table 3.6: *Solved authentication scenarios*

3.3.3 Unsolved scenarios using a CA based approach

From the results of the review, it can be observed that even though CA based approaches are widely accepted as a method for group membership, not all possible authentication scenarios are directly solved with current proposals. This subsection presents how to modify current proposals in order to solve all remaining scenarios under this popular approach.

Scenario 3As can be solved by limiting the access to the CA certificate (and the contained public key) only to members of Γ^R. In this way, only this set of peers hold the necessary information to authenticate other peers. However, it must be considered that using this solution, Γ^R does not accept arbitrary resignations, since once a member has accessed the CA's root certificate, the only way to deny its usage would be to change the CA's private key. As a result, this solution

should only apply to quite static Γ^R sets.

Using a CA based approach is specially constraining for collaborative authentication scenarios. In order to solve them it is necessary to provide additional capabilities to the base schema. One possibility would be using byzantine [Lam82] authentication, so an agreement must be reached in order to consider any peer as a group member. In fact, a similar approach is presented in [Zho02]. However, the use of this kind of systems only contributes from a fault tolerance perspective. The proposed approach does not really force peers in Γ^A to collaborate, since all of them have the necessary data in order to fully provide authentication.

In order to truly solve collaborative scenarios from a mechanical standpoint, not a fault tolerance one, peers in Γ^A must collaborate, or otherwise be unable to provide authentication, just as it happens for the case of registration in scenarios 2Rc or 3Rc. This is achieved by extending the threshold system to the CA's public key across Γ^A, so all of the authenticating peers must work together in order to validate any peer group certificate.

The most problematic scenarios for a CA based approach are the ones where only those peers which took part in the registration process, Γ^R_B, may authenticate (scenarios 4As y 4Ac). Since the information related to Γ^R is common to all their members, and no unique element exists for any of them, it is not possible to overcome this limitation. As a result, these scenarios cannot be solved even by modifying the base schema.

CA based approaches are very focused towards collaborative registration scenarios and single peer authentication ones. However, it is possible to modify some aspects of its behavior so they are able to satisfy some of the listed scenarios. It is also important to take into account that some strong restrictions may appear when some group member resigns from Γ^R, meaning that this approach is not feasible in highly dynamic scenarios.

3.4 Solving scenarios via web-of-trust

In a P2P environment, the web-of trust approach could be considered ideal, since it maximizes peer autonomy and equality. No special authority, be it internal or external to the group, embodied as a CA, exists. For that reason, it is worth studying each of the proposed scenarios under this approach.

Even though the review in Section 3.3 seems to point out that a web-of-trust

based approach is only useful in a very limited number of scenarios, it may be possible to solve most of the scenarios with the adequate modifications to current proposals.

3.4.1 Registration scenarios

Web-of-trust approaches focus on providing maximum peer autonomy. As a result, they can be directly applied to single peer registration scenarios (2Rs and 3Rs).

In order to solve scenario 2Rs, two sets of peers must exist, those who may register and those who cannot. These sets are created via the coexistence of two different kind of trust relationships within the web-of-trust (generated, for example, by adding additional properties): trust relationships between two Γ^R members and between a member of Γ^R and a peer which is not a member. Those peers which have become group members via the second kind of trust relationship may not register group members. Rogue behaviors can be easily detected when validating the trust path in the following manner: checking that such peer has some trust relationship from a peer which is a group member because of the latter kind of relationship.

Scenario 3Rs is directly solved, as summarized in Table 3.5, being the standard web-of-trust operation mode.

Collaborative scenarios are solved by requiring that t group members trust the candidate. The registration process is not considered complete until t different trust relationships have not been generated. Additionally, this approach makes it possible to complete the registration process in a totally asynchronous manner, which is a very useful aspect in a P2P system. The only difference between scenarios 2Rc and 3Rc resolution falls in whether it is considered that $\Gamma^R \subset G$ or $\Gamma^R = G$.

3.4.2 Authentication scenarios

The strong differences between CA based approaches and web-of-trust based ones become evident in the authentication scenarios. In the web-of-trust model, collaboration may be enforced by keeping the information needed for the authentication process distributed across group members. As a result, single peer scenarios are the ones which must be specially modified in order to be solved, by properly distributing information which is originally tied to arbitrary peers.

Single peer scenarios:

The most difficult scenario under a web-of-trust approach is 2As, since all information regarding trust paths must be stored in every peer. As a means to solve the scenario, it is necessary that all peers in G keep a repository with the minimum number of trust relationships that allow them to reach other members of G (for example, calculated using the Dijkstra algorithm). This option is equivalent to the one used in OSPF's [osp98] routing mechanisms.

Scenario 3As is a more restrictive case, since it is necessary that only members of Γ^R keep the necessary information in order to generate a trust path. Solving this scenario implies using the same system as the one proposed for scenario 2As, but moving from the general scenario to a more restrictive one, considering $\Gamma^R = G$.

To sum up, scenarios 2As and 3As may only be properly solved in groups with a low cardinality of G and low dynamic behavior.

Scenario 4As, by being the most restrictive one, can be trivially solved without any impact on peer autonomy.

Since Γ^R_B is exactly the set of peers which generated B's trust relationship in order to join the group, it is very easy for any of them to authenticate B. Peers in Γ^R_B hold all the necessary information as they were the creators of such trust relationship. However, in order to truly limit to Γ^R_B the capability to authenticate, it is necessary that they also keep their public key secret, so no other group member may validate signatures generated by them.

Collaborative scenarios:

Scenario 2Ac is already directly solved, as shown in Table 3.6, since it is the base case for a web-of-trust approach: any peer may try to retrieve the trust path by collaborating with other group members.

Scenario 3Ac can be solved if peers within Γ^R do not collaborate with those outside of Γ^R. Since only peers in this set may register, it is obvious that all trust chains must go across some member of Γ^R at some step. If it is impossible to retrieve information from this set of peers, no valid trust chain may ever be retrieved. Only peers in Γ^R will be able to retrieve complete trust chains.

For the most restrictive scenarios, 4Ac, it is important to remark that it makes no sense unless it is linked to a collaborative registration scenario, since it must

be true that $|\Gamma_B^R| > 1$. A solution may be reached if members in Γ^R do not collaborate with any other peer in order to retrieve trust paths (in this case, both members or not of Γ^R) and, additionally, they all keep its public key secret. At this point, since it is impossible to retrieve trust paths with a length greater than 1, the only way for B to be authenticated is by applying to those peers in Γ_B^R.

Once this restriction has been established, it is possible to establish an agreement protocol between members of Γ_B^R (such as a byzantine system) in order to finally authenticate B in a collaborative manner. It must be noted, however, that in this case the agreement protocol is not used to produce the same task several times (in order to provide fault tolerance), but each member of Γ_B^R is solving a different task: validating its own trust relationship to B. As a result, in this case it can be considered licit to use a byzantine agreement protocol (in contrast with the one described in subsection 3.3.3), since it is truly forcing collaboration.

From the proposed solutions, it can be concluded that, with appropriate modifications, it is possible to solve most of the scenarios with a web-of-trust based approach. In contrast with the widely used CA based approaches, a web-of-trust one is specially suited for distributed environments where peers join efforts in order to provide network services, such as it is the case in P2P environments. In fact, some of the most difficult scenarios for a CA based approach are easily solved using a web-of-trust approach.

3.5 Chapter Summary

This chapter has presented an exhaustive description and classification of the different possible scenarios for group membership and access control in a P2P environment. The registration and authentication steps have been defined in order to classify such scenarios according to the involvement and roles of group members. Furthermore, the main properties of every scenario have been discussed, from a performance (availability, responsiveness, misbehavior tolerance,...) and peer equality standpoint. Finally, we have provided a review on how each scenario may be solved using both a CA based approach and a web-of-trust based one, highlighting which scenarios cannot be properly solved in an efficient way.

Chapter 4

A web-of-trust based group membership model

In Chapter 3 we have presented the classification and solution for different group membership scenarios. It has been pointed out that, even though the most popular approach to group membership is a CA based one, using web-of-trust as a means to peer group membership is feasible, since it applies to a broad number of those scenarios.

This chapter proposes a secure and scalable method, based on the concept of web-of-trust, to provide group access control and to check group membership [AM06b]. The approach takes into account the nature of P2P networks, being fully decentralized and paying special attention to the autonomy of its group members and their self-organization. Each peer should initially manage information only directly related to itself in a manner that all necessary services are provided by its members, avoiding dependency from external group entities.

The main contribution of this approach is its ability to adapt to a broad range of group policies and group membership scenarios, providing necessary capabilities to be more restrictive or open when needed. Furthermore, the proposal minimizes the length of certification paths in order to avoid the validation of long certification chains, which is the main problem in several of the current methods.

This chapter is organized as follows. In Section 4.1, the proposal for peer group membership access is described. We define the peer group access model and all

the necessary membership services to allow group membership in a web-of-trust environment. In Section 4.2, the model is enhanced in order to provide some degree of anonymity using ring signatures. First, the concept of ring signature is explained and, following, we present how it may be applied to the proposal. A brief security analysis closes this section. Finally, Section 4.3 summarizes the main highlights of this chapter.

4.1 Peer group access

As already introduced, the basic concept of web-of-trust in PGP is ideal for a P2P environment, where all peers are autonomous and share efforts in order for the network to continue to operate, without any kind of centralized infrastructure. However, since P2P networks are fully distributed environments where it is not possible to rely on a single specific peer, central certificate repositories cannot be directly used, as PGP proposes. If peers have to be really autonomous, each one should manage only its own trust relationships and will need to collaborate and aggregate resources in order to infer trust relationships.

For the rest of this section, the definitions provided in Section 3.1 on regards to peer group registration and authentication stand.

4.1.1 Group membership trust model

Authentication in a web-of-trust depends on a path of trusted intermediaries that will travel from the authenticating peer to the one to be authenticated. These intermediaries are defined as *introducers*. When a peer trusts an introducer, it implies a certain confidence on the capability of the introducer to provide faithful certificates. If a path really exists between both peers, the identity may be confirmed via the evaluation of the trustworthiness of each introducer's key. Otherwise the user's identity cannot really be confirmed and no real information is obtained from the web-of-trust.

The different trust relationships in a web-of-trust are usually represented as a directed graph, where edges represent trust relationships, as shown in Figure 4.1. Here, for example, Z trusts Y, which, in turn, also serves as an introducer for both X and U. That means Z may authenticate X and U through Y.

In order to use web-of-trust concepts in our model for group access control,

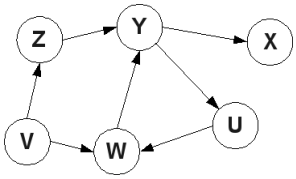


Figure 4.1: *Web of trust representation*

trust relationships are translated to proof of group membership. Whenever peer A creates a trust relationship with peer B, it is acting as an introducer to B for the rest of group members. This means that A is vouching for B's group membership to other group members. In the case presented in Figure 4.1, Z acts as an introducer for Y, which then also acts as a introducer for X and U.

In our model, two different sets of peers exist within a peer group: one composed by peers that belong to Γ^R (those which are allowed to register new members), and another composed by the rest of group members $G \setminus \Gamma^R$. At this moment, no initial assumption is made regarding the cardinality of both sets or how a peer becomes part of each set, but it is obvious that it is necessary that $\Gamma^R \neq \emptyset$ in order for new peers to be able to join the group.

Since two different sets of peers exist within G, two different types of trust relationships are distinguished between peers, depending on which set both peers belong. Both types of trust relationships are created in the same way than in a standard web-of-trust, by signing the trusted peer public key, generating a certificate with a specific expiration date. The created certificate specifies which kind of trust relationship it is for. Both trust relationships reinforce the fact that a peer is member of the group.

Patron relationships are established from peers which belong to Γ^R to a peer which does not belong to Γ^R. These trust relationships are unidirectional, just like the typical web-of-trust relationship. When peer A signs peer B's key, it will be considered to be its *patron* within the group.

Backbone relationships are established between peers which both belong to Γ^R, but are considered to be bidirectional. In order to create, backbone relationships, both peers have to sign each other's public key.

An example of these different sets of peers and types of trust relationships is shown in Figure 4.2. Grey peers are those which belong to Γ^R, and white ones

do not belong. Backbone relationships between peers are marked as bidirectional arrows, whereas patron relationships are marked as unidirectional arrows.

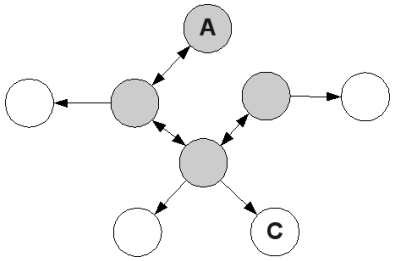


Figure 4.2: *Initial group membership trust graph*

It must be taken into account that in a global overlay network different peer groups must transparently operate. Under this model, each peer group manages membership with its own web-of-trust, which means that some number of independent webs of trust will coexist. Such coexistence of different webs of trust within a single network is achieved by binding each signature to the specific peer group the patron peer is vouching for, and will only be valid for operations regarding that specific peer group. In order to bind each signature to a particular peer group, the group *id* is included into the signature. A peer may hold different signatures from the same patron, but each one bound to different peer groups.

Since peers are autonomous, each one exclusively manages only information related to itself. This means that each peer will have information related to:

- Its own private/public key pair.
- The signatures from other peers.
- The list of peers trusted by him.

Any other information the peer would need must be provided by other group members.

PGP defines several degrees of trust (none, marginal and complete), and some proposals for path validation in ad hoc networking take into account trust evaluation and focus on how to assign specific values to trust [Edi04; Zha04; Bet94]. However, in our proposal, only two different degrees are initially considered: whether the relationship exists or not (0 or 1 respectively).

4.1.2 Group membership services

Once the basic trust model has been described, we detail how to provide the basic group membership services. A list of group management services and dynamics may be found in [Zou05]. However, such approach does not identify the difference between group registration and authentication. Next, services for single operations are described, since bulk operations can be reduced to an aggregation of such simple operations.

Group setup

In the case that a single peer decides to create a new group, no trust relationships are needed, and the peer decides whether it will be part of Γ^R or not. It must be noted, however, that in the latter case the group will never be able to grow, so it is an impractical decision.

When a set of n peers want to create a group, the group setup process can be divided in two phases. In the first one, an initial peer does belong to Γ^R. In the second phase, the rest of $n-1$ peers perform membership registration, explained as follows.

Membership registration

When a new peer, B, wants to register to the group, it must apply for membership to any peer which belongs to Γ^R, such as A. If agreement is reached, a patron relationship is established and A becomes B's patron for this group. The model does not impose any restriction on deciding why a new peer is accepted into the group. This decision relies on the group policies or the individual decision of A, and it goes beyond the scope of the model.

Since signatures expire, B should renew its relationship with some patron (usually, the same as in the initial registration) when the expiration date nears.

The model also accommodates to the possibility that n peers may become patrons of B. Under this assumption, such relationships may be created once B is already part of the group or at the precise moment of group registration. This assumption enables the group to provide redundancy in order to solve possible availability problems as it is shown in Section 4.1.2. Furthermore, this enables implementing stricter group policies or scenarios where a minimum number of patrons must be collected before B is considered a group member. Under this

policy, group membership cannot be achieved with a single patron, reinforcing security at the cost of a less agile registration procedure.

Change of role

At some time, any peer may decide to become part of Γ^R. On this regard, the model does not enforce any rules for the creation of backbone relationships, imposing no restrictions on group behavior at this level. Since P2P environments encompass a completely open network where peers self-organize and collaborate in order to achieve basic operations, any peer which is part of the group may apply to become part of Γ^R at any time, just like it may apply to join the group. Group policies or strategies should decide if this peer is granted such role change. In fact, it may be completely possible that $G = \Gamma^R$, fulfilling total peer equality. As peers which belong to Γ^R perform more operations, it is expected that only those nodes with sufficient computer power or bandwidth will apply.

It may also be possible that some peer D decides to leave Γ^R for some reason. This process is summarized in Figure 4.3. First of all, D must announce those peers under its patronage its intention so they can find new patrons (in the case that each peer that does not belong to Γ^R only has one patron). Until that happens, D cannot leave Γ^R. Otherwise, some group member could not be able to authenticate to the group.

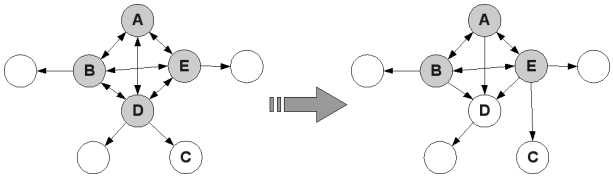


Figure 4.3: *Change of role from* Γ^R

Then, D must also tell each of those peers with a shared backbone relationship (peers A, B, E in Figure 4.3) it will leave Γ^R. All such peers are known to him, since each peer keeps the list of peers which trust him in the form of signed certificates. All of them now become its patrons, changing their relationships. If some peer is currently off-line, for example B, D will wait until its certificate signed by B expires, or until B reconnects (while B is offline, the fact that there is a backbone relationship pending to be revoked does not affect the system,

since B will never interact with the group). Since in backbone relationships both certificates are created at the same moment, that means that the certificate signed by D that B keeps will also be expired, so the relationship is completely over. At that moment, D no longer needs to try to contact B.

In the case that B is the only peer D has a backbone relationship with, D cannot leave Γ^R unless D wants to lose group membership.

Membership authentication

In order for a peer C to provide proof of membership, a trust path must be found between the authenticating peer and C. Both kinds of trust relationships are eligible when searching this path. However, this trust path must accomplish the additional condition that all contained signatures must be bound to the peer group C is trying to access. In Figure 4.4 a case is shown where A may be able to correctly authenticate C as a group member.

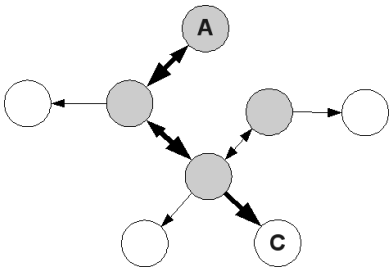


Figure 4.4: *Finding a trust path between two peers*

This model includes the possibility that some peers, those in $G \backslash \Gamma^R$, do not have sufficient information to authenticate, even in the case that all peers are online. In this way, the model has the capability to be adapted to specific group policies that need such restriction.

In order to minimize the length of certification chains in the proposed model, when A successfully authenticates C, A checks whether it shares a backbone relationship with C's patron. If such relationship does not exist, a new one is automatically created. As the group life progresses, peers which belong to Γ^R will eventually create a connected graph. From that moment, any peer in Γ^R may authenticate any other peer in G in two hops.

The evolution of the backbone relationship graph over time is shown if Figure 4.5. In *(a)* an initial state is presented. When A authenticates C, a new backbone trust relationships is created *(b)*. Eventually, full connectivity is achieved in *(c)*.

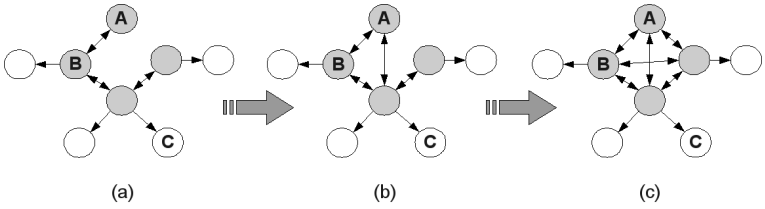


Figure 4.5: *Complete backbone graph progress*

The creation of multiple patron relationships is important in order to avoid those cases where authentication might fail because insufficient peers are online, a key issue in P2P and ad hoc networks. For example, in Figure 4.5 case *(a)*, whenever B becomes unavailable, it would be impossible for C to provide proof of group membership to A, since a trust path could not be retrieved. This problem is solved in cases *(b)* and *(c)*, via the eventual growth of the backbone relationship connected graph.

4.2 Unlinkability in peer group membership

In Section 4.1 we have just shown that it is possible to solve the different peer group membership scenarios presented in Chapter 3 using a web-of-trust based approach. However, there are other desired security features in group based P2P systems. One of such features is anonymity: allowing users to join and interact within a peer group without exposing their identity, protecting their privacy and escaping censorship. In such scenario, proof of group membership needs additional mechanisms which preserve anonymity.

As far as P2P systems are concerned, currently, several initiatives exist to provide an entirely anonymous network. The most popular ones are Freenet, FreeHaven or Tor [Din04]. In such systems, it is not possible to easily guess the source of messages transmitted across the network. A general survey on anonymity in P2P systems may also be found in [RY08]. Unfortunately, the aforementioned initiatives do not take into account peer group based environments or the different

roles which peers may take according to group membership scenarios. These proposals rely on the assumption of a flat network where peers do not from groups. Anonymity is exclusively focused on messaging, and resource access and publication.

The goal of this section is to extend the group membership method presented in this chapter with some degree of anonymity, allowing peers to prove membership to each other without disclosing their actual identity. Specifically, the proposed method [AM09a] is concerned with identity unlinkability: even though members of a peer group may know the identity of its current members, it is not possible to trace authentication attempts to a specific identity or tell which have been initiated by the same peer. This is achieved with the help of ring signatures [Riv01; Ren08].

The proposal assumes that peers are already provided with anonymous transport, having the capability to anonymously exchange messages at a lower layer using any of the initiatives that already exist, such as [Cla02; Din01; Din04]. Otherwise, the point of anonymously proving peer group membership becomes moot as the source peer identifier is sent across the network in clear text.

4.2.1 Ring Signatures

The notion of a ring signature scheme [Riv01] is related to that of group signature [Cha91]. In the latter, a trusted group manager predefines certain groups of users and distributes special keys to their members. Each member can use these keys to anonymously generate signatures which look indistinguishable to other group members. In contrast, the former does not need a group manager. This is a highly desirable feature in a self-organized environment such as P2P.

Ring signatures are useful when the members are autonomous and do not want to rely on other peers. They are signer-ambiguous and provide no way to revoke the anonymity of the actual signer. It is only necessary to assume that each peer group member already holds a private/public key pair of some standard signature scheme. A ring signature is generated by the actual signer declaring an arbitrary set of possible signers, which includes himself, and computing the signature by himself using only his private key and the other members' public keys.

In this scheme, the set of possible signers is called a ring. The ring member who produces the actual signature is the signer and each of the other ring members is

a non-signer. The signer does not need the consent or assistance of the other ring members to put them in the ring, only knowledge of their public keys is needed. Two procedures are defined:

- $\sigma = ring - sign(m, PK_1, PK_2, ..., PK_n, SK_i)$ produces a ring signature σ for the message m, given the public keys $PK_1, PK_2, ..., PK_n$ of the n ring members, together with the private key SK_i of the i-th member, the actual signer, for $1 \leq i \leq n$.
- $ring - verify(m, \sigma)$ accepts a message m and a signature σ and outputs either true or false. σ includes $(PK_1, PK_2, ..., PK_n)$, the public keys of all possible signers.

Verification satisfies the usual soundness and completeness conditions, but since ring signatures are signer-ambiguous, the verifier is unable to determine the identity of the actual signer: in a ring of size n, probability is not greater than $1/n$. This limited anonymity is unconditional [Riv01], since even an infinitely powerful adversary cannot link signatures to the same signer.

Ring signatures are also particularly efficient, since generating or verifying a ring signature costs the same as generating or verifying a regular signature plus an extra multiplication or two for each non-signer. This means that the scheme is still useful even when the ring cardinality is very high.

4.2.2 Authentication unlinkability with ring signatures

The following notation will be used in this section.

- PK_i: Peer P_i's public key.
- SK_i: Peer P_i's secret (private) key.
- $Cert_i$: One of peer P_i's certificates, containing PK_i. It must be noted that in a web-of-trust, P_i may hold several certificates (obtained from different patrons). However, they all always contain PK_i.
- $E_{PK_i}(x)$: A string x encrypted using the public key of peer P_i.

Group access control using asymmetric cryptography by means of digital certificates, is normally performed through trust paths. When some peer P_n

wants to prove to some other peer P_1 that he is part of the group (P_1 authenticates P_n), all certificates conforming a trust path between P_n and P_1 are transmitted. A trust path from peer P_1 to peer P_n is a list of certificates $(Cert_1, Cert_2, Cert_3, ..., Cert_n)$, where P_1 signed $Cert_2$, P_2 signed $Cert_3$, etc. up to $Cert_n$. Consequently, P_n may be considered a group member by P_1 only if such trust path exists.

However, using this approach, the public key PK_n becomes the main constraint in order to achieve unlinkability between authentication attempts. Even in the case that $Cert_n$ does not contain any information regarding P_n's identity, the public key can be regarded as a unique identifier or pseudonym and only some degree of pseudonymity is ultimately achieved.

In order to solve this problem, we introduce the concept of *trust tree, TT* between two peers, which will be used as a means for authentication instead of a trust path, in order to solve this issue.

A trust tree between peers P_i and P_j, TT_j^i, is defined as a set of certificates $\{Cert_1, Cert_2, \cdots, Cert_n\}$ with the following properties:

- $\exists i \in \{1, \cdots, n\}$ such that $Cert_i$ is P_i's certificate.
- $\exists j \in \{1, \cdots, n\}$ with $j \neq i$ such that $Cert_j$ is P_j's certificate.
- $\forall k \in \{1, \cdots, n\}$ there exists a trust path from $Cert_i$ to $Cert_k$. As a result, it is guaranteed that a trust path exists from $Cert_i$ to $Cert_j$.

The certificate of P_j in TT_j^i is unknown to anybody but its creator: P_j. It is only guaranteed that PK_j is in some certificate within TT_j^i. Also note that, in contrast with trust paths, $Cert_j$ does not need to be the edge one, it may be anyone within TT_j^i. As a result, given a peer group, more than one possible TT_i^j may exist between two peers.

TT_j^i can be envisioned as a certificate hierarchy which comprises several trust paths, as shows the example in Figure 4.6.

Since TT_j^i is a set of certificates, it contains the set of public keys $PK_{TT_j^i} = PK_1, \cdots, PK_i, \cdots, PK_j, \cdots, PK_n$. As a result, it is feasible to apply a ring signature scheme where P_j may specify the TT_j^i as the set of possible signers. Choosing a ring signature scheme is worthwile in a P2P environment as shown in Section 4.2.1.

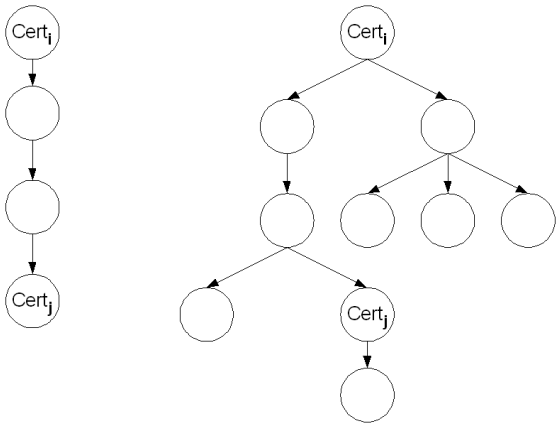


Figure 4.6: *Trust Path and possible Trust Tree between peer P_i and P_j*

4.2.3 Authentication process

In order for P_j to anonymously proof group membership to P_i, a valid TT_j^i must be generated by P_j in advance. The authentication process then follows, as shown in Figure 4.7.

1. A session identifier, *sid*, is chosen by P_j and sent encrypted to P_i.
2. P_i generates a pseudorandom nonce, r, which is sent in response to P_j. P_i stores both *sid* and r as an authentication transaction in progress.
3. P_j generates σ, a ring signature of r using its own private key and the set of public keys in TT_j^i.
4. P_j sends to P_i the values *sid* and σ, which includes both the signature algorithm result and its related set of public keys, $PK_{TT_i^j}$.
5. P_i uses *sid* to identify the transaction, retrieves r and checks the signature's correctness.

If $ring - verify$ validates, P_i checks the validity TT_j^i, as it will be shortly explained. In case that TT_j^i is valid, P_j's peer group membership is considered proven to P_i. Short term access as a group member is granted using $E_{PK_i}(sid)$ in any further messages. During this access, interactions are linked via *sid* (but

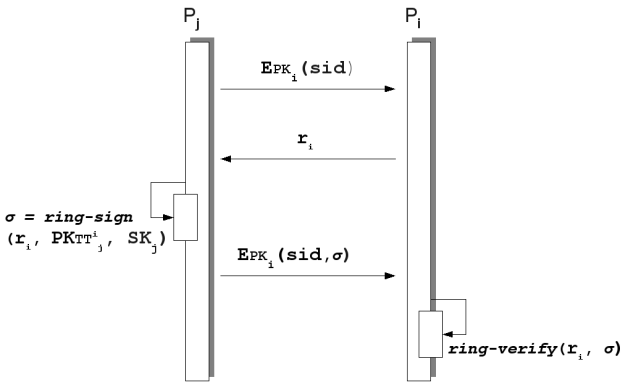


Figure 4.7: *Peer group membership authentication protocol*

the identity of P_j is never disclosed). However, P_j may reset the identifier by re-authenticating. Any authentication attempt remains unlinked to the previous ones.

In order to check the validity of a trust tree TT_j^i, for any certificate from some peer $P_k \in TT_j^i$, the trust path from P_i to P_k must exist. The TT validation process is considered valid if all trust paths between $Cert_i$ and any other certificates are valid, which means that any subject within the TT is a peer group member. During this validation process, P_i stores in a local cache correct trust paths. Just certificate subjects are stored, not the whole certificate. This cache can be used in further TT validations in order to speed up the process by first looking up if a trust relationship between P_i and some other peer has already been checked in previous TT validations. An advantage of keeping this cache is the fact that it can be used in the validation process of any TT within the peer group.

In this proposal, each possible signer for a given TT (each certificate subject within the TT) becomes the anonymity set [Pfi01] for each authentication attempt. The signer's identity becomes hidden within all subjects in the TT, however, since during the validation process it has been guaranteed that all subjects within the TT are group members, it can also be guaranteed that the signer is a group member.

A TT is used instead of a simple trust path in order to both dynamically expand the cardinality of the anonymity set and avoid clearly pinpointing the

actual signer (which is usually the owner of the edge certificate). Using a standard trust path as a ring signature signer set is not desirable since its cardinality, once established, will usually remain static. Since the degree of unlinkability relies on the number of possible signers, in some instances, such cardinality may become too narrow to be considered acceptable. TT's take advantage of some other peer's long trust paths in order to increase the cardinality of the signer set. Furthermore, at each authentication attempt between P_i and P_j, the later is not bound to always using exactly the same TT. P_j is able to generate different TT's which are considered valid, resulting in diverse anonymity sets which may be used at authentication attempts.

Even in the worst case scenario where P_i directly trusts P_j, an acceptable TT may still be produced. It is enough to include some other peers which conform trust paths from A, even though B does not appear in those paths, as shown in Figure 4.8 as an example. Even in this scenario it is still possible to chose which is the cardinality of the anonymity set, up to the full peer group's cardinality.

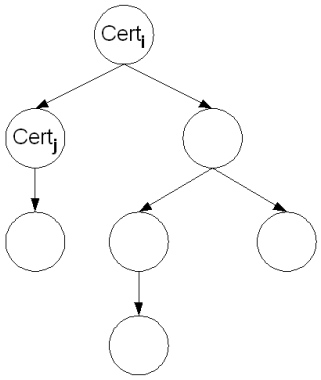


Figure 4.8: *Possible TT_j^i where P_i directly trusts P_j*

Trust tree generation

A TT cannot be generated by arbitrarily collecting and setting together member public keys, which is the assumption in a basic ring signature approach, since it must be ensured that all the included public keys conform some trust path from the authenticating peer. No unconnected certificate may be included.

There are several methods to obtain the necessary certificates in order to generate a TT by taking advantage of peer group membership operation based on web-of-trust. However, it is highly desirable that certificate retrieval is performed along standard network operation, since the impact of using TT's on network performance is minimized, and most important, other peers then cannot know whether certificate retrieval is being requested in order to compose TT's or just in order to routinely operate.

Feasible methods for some peer P_j to retrieve certificates from other group members, other than just directly requesting them, are:

- Whenever P_i is requested a certificate by someone interested in becoming a group member, acting as a patron following the approach in [AM08a]. The generated certificate may be stored for later use for TT creation. An advantage of this option is that all certificates will be useful for TT generation, since all of them conform a valid trust path from P_j. Using this method is also an incentive to become a patron, since the most certificates P_j generates, the easier it is for him to generate large anonymity sets.

- In web-of-trust environments, peers already know its patrons' certificates.

- In onion routing anonymous networks, which are the most popular ones, the sender establishes the message path by retrieving the certificate of each peer in the desired path. This is necessary to create each message encryption layer. That means that some method to directly retrieve other peer's certificates must exist. An advantage of this method is that both onion routing operation and TT generation make use of the same information.

As shown at the beginning of the authentication process in subsection 4.2.3, TT's may be generated in advance. It is not necessary to be create them at the precise moment just before initiating authentication. Peers may slowly generate a large TT as they operate within the peer group and learn about new certificates. Different subsets from a large TT, which could amount to the whole peer group in the best case scenario, may be used at each authentication attempt, as shown in Figure 4.9. There are two main reasons for not using the whole information and just using subsets: avoiding sending large TT's across the network, which may impact performance and slow TT validation, and preserving the security of the scheme, as will be explained in Section 4.2.4.

Even though a large TT can be constructed along standard peer operations within the group, it is important to take into account that such TT must be maintained coherent with the actual status of the group. Resignations or trust relationship revocation may change the status of actual trust relationships. Therefore, trust relationship status must be updated every some time interval. This is only necessary for trust relationship where the peer which stores the TT does not participate (the peer is neither patron nor trustee). In that case, the peer already has an accurate knowledge of the trust relationship's status, being part of of the relationship.

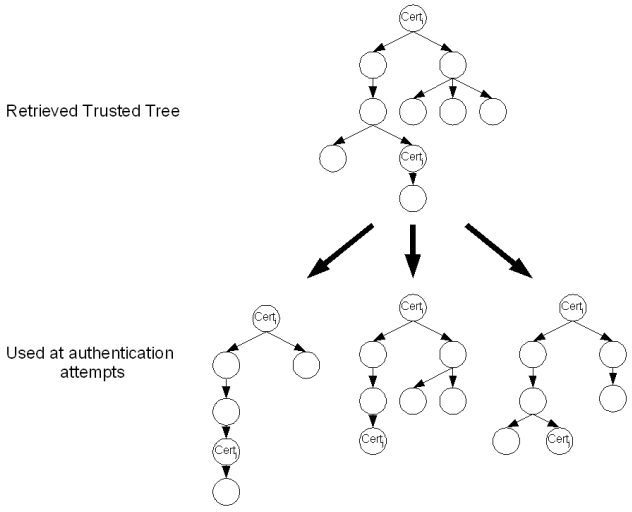


Figure 4.9: *Using a large TT to generate smaller TT's*

4.2.4 Security analysis

In this section attacks on anonymity are analyzed. This analysis will focus only on the authentication process, since an anonymous transport method is assumed. For that reason, all strengths and weaknesses of the anonymous networks will be inherited. Common attacks in anonymous networks such as the predecessor attack [Wri04], for example, are not discussed, being completely concerned with message relay.

- *Trusted tree reuse:* A big concern in this proposal is the reuse of TT's. If the same TT is always used, it will ultimately become like an identifier and it will be trivial to link different sessions. For that reason, the same TT should not be reused in different sessions. This problem is solved with the generation of a bigger TT and then only using smaller subsets.

  However, the proposed scheme still minimizes this attack to some degree, since it cannot be guaranteed that the same trusted tree, received several times by the authenticating peer at different group membership authentication attempts, was sent from the exact same peer. Two peers may generate the same TT, and still follow the properties listed in Section 4.2.2, since a given TT may be used by any peer whose public key is contained within.

- *Timing attack:* In a timing attack, an authenticating peer may try to link different sessions by comparing response timings in the authentication process (message round trip time). Although anonymous networks already take this attack into account, this problem can also be solved by purposely delaying responses a random amount of time in the authentication protocol.

- *Intersection attack:* This attack is complementary to that of TT reuse. The authenticating peer may try to compare different TT's used by the same peer and intersect all of them, trying to decrease the number of suspects. An underlying anonymous network does constrain this attack only to the authenticating peer (it cannot be attempted by a middle point peer). However, this attack is countered since it is not possible for an attacker to know which TT's come from the same peer in order to compare them. All authentication attempts (as well as sessions) are completely independent and no information is sent along each one in order to relate different attempts to each other. Given a set of different TT's in which an specific public key appears in all of them, it cannot be guaranteed that some of them came from the same peer.

4.3 Chapter Summary

This chapter proposes a method for access control in peer groups using a web-of-trust approach, which takes into account the main properties of peer equality and decentralization in a P2P environment. The method may be adapted to a broad

range of group policies and group membership scenarios, such as those presented in Chapter 3, providing necessary capabilities in order to be more restrictive or open when needed. Furthermore, the proposal minimizes the length of certification paths in order to avoid the validation of long certification chains, which is the main problem in several current methods.

Once the base method for group membership has been established, it has been extended with unlinkability between authentication attempts. This is achieved by using ring signatures and introducing the concept of trust trees as the mean to transport public keys as well as providing an anonymity set for the signer. Trust trees may be constructed according to each peer's own needs and may be changed over time. This unlinkability approach nicely meshes with the proposed method for peer group membership since the link between trust tree size and the anonymity set directly rewards those peers who decide to act as patrons within the group, providing an incentive to peers which act as such. In fact, in a web-of-trust based environment, it is very important that as many peers as possible agree to act as patrons.

Chapter 5

A group membership specification under JXTA

The peer group membership model presented in Chapter 4 is completely generic and may be applied to any P2P framework which requires support for secure groups. In this chapter, a complete specification of this model is presented for a particular P2P framework: JXTA. The idiosyncrasies of the JXTA platform are taken into account in order to create a system which is fully compliant with JXTA's specification. Specifically, its core services are used to provide group member registration and authentication [AM08a].

The reasons for choosing JXTA are twofold. First of all, as it has already been introduced, JXTA [SUN01] defines the concept of *peer group*: a collection of peers with a common set of interests. This concept is one of the main foundations of the JXTA architecture and is prevalent throughout all its specification [SUN07]. Thus, we consider JXTA an ideal testbed for a secure approach to peer group membership. In addition, as introduced in Chapter 1, JXTA has gathered a great popularity in the recent years, becoming one of the main frameworks for the development of P2P applications, at both an academic and enterprise level. As a result, it can be considered a mature P2P technology with many contributors.

This chapter only introduces the most basic JXTA concepts in order to understand how peer group membership is achieved. A detailed description of the availiable security mechanisms in JXTA is provided in Chapter 6.

This chapter is organized as follows. Section 5.1 presents a detailed description of the generic JXTA membership and group access control framework. Section 5.2 describes the group membership specification details, explaining how the Membership and Access Services are deployed using the JXTA framework and which are the required support services. Section 5.3 Summarizes this chapter.

5.1 An overview of group membership in JXTA

JXTA manages group membership, through two different services: the Membership Service and the Access Service. Both are core services which make use of the base JXTA protocol specification in order to achieve their ends. The Membership Service manages identities within a peer group, providing each group member a *credential*. Peers may include this credential in messages exchanged within a group in order for other member to know who is making a request. With this information, the JXTA Access Service evaluates the credential when a service is accessed and decides whether the request will be granted or denied.

5.1.1 Membership Service

The Membership Service allows every peer to establish its identity within a group and obtain a credential by successfully completing the *join* process. This process is divided in three distinct steps: setup, application and validation, as depicted in Figure 5.1.

- In the *setup* step, the peer chooses which authentication method will be used for the whole process. An AuthenticationCredential is chosen and sent to the Membership Service via the *apply()* method. Some examples of specific authentication methods are interactive authentication (via a GUI), plain text authentication (programmatical), etc. The AuthenticatorCredential can also be used as a way to identify and allow interaction with the peer even before it becomes a group member. If all parameters are correct and the choice is feasible, the peer receives an Authenticator from the Membership Service.

- Following, in the *application* step, the peer completes the Authenticator with all necessary information and tests its correctness with the *isReadyforJoin()*

method. It will not be possible to join the group until the Authenticator is correctly initialized.

- Finally, in the *validation* step, joining the group is possible if the Authenticator is correct. The Membership Service checks whether the peer may assume the claimed identity and creates a credential. Since the join method is provided by each implementation of the Membership Service, the identity ownership check may be as complex as necessary.

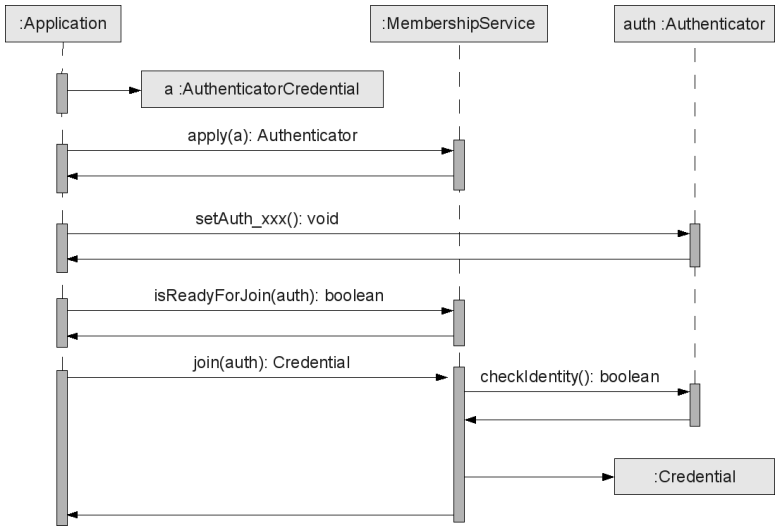


Figure 5.1: *Peer group join process*

Before a peer may join a group, it must be authenticated by providing a correct Authenticator to the Membership Service. An Authenticator contains all the required information in order for the Membership Service to check that the requested identity can be granted. Each different Membership Service specification provides its own definition of Authenticator, suited to its needs and inner workings.

In case that a peer decides to give up membership to a specific group, a *resign* method exists. With this method, the credential generated in the join process is discarded. Group resignation is voluntary, the Membership Service does not support active membership revocation triggered by other members.

5.1.2 Access Service

Credentials generated in the join process may be sent whenever a group service is accessed, as part of protocol exchanges. The JXTA Access Service provides mechanisms in order to check them, allowing services to decide whether access should be granted or not.

The sequence diagram for the Access Service is shown in Figure 5.2. A single operation is offered to the peer group in order to check a credential for a privileged operation.

Figure 5.2: *Access control check via Access Service*

The possible Access Service results are DISALLOWED, PERMITTED, PERMITTED_EXPIRED (the operation would be permitted but the credential has expired) and UNDETERMINED.

It is very important to remark that the Access Service is fundamental in order to deploy any group access control model under JXTA. The reason is that JXTA allows a peer to directly instantiate any group and access its published services without credentials (using a "nobody"-like default identity). The use of credentials is optional. As a result, unless the use of credentials is enforced via the Access Service, the process of joining a group via the Membership Service and obtaining a proper credential becomes pointless.

5.1.3 Existing service specifications

The JXTA reference implementation, as far as version 2.5 [jxt07], provides three Membership Services: *None*, *Passwd* and *Personal Security Environment (PSE)*.

The *None* Membership Service is intended for peer groups which need no authentication. Since any peer may claim any identity, it is recommended that credentials should only be used within the group for purely informational purposes.

The *Passwd* Membership Service relies on a Unix-like username and password

pair for peer authentication. In order to claim an identity, the correct password must be provided. The list of pairs (username and password) is distributed to all group members, which means that the password file equivalent roams freely through the overlay network. This group membership service was created as a sample and a means for testing, since it is completely insecure. For that reason, it is advised in the JXTA documentation that it should never be used in any serious application.

The default Membership Service is *PSE*. This service provides credentials based on PKIX [CCI88] certificates. However, it must be taken into account that the Authenticator for this service exclusively relies on passwords: the peer is authenticated to the group by being able to access the keystore which holds its private key. This means that the join method itself is not based on digital signatures or certificates, just the credential format itself. In fact, as presented, PSE is more of a kind of toolbox that allows the implementation of different models based on securing identities via digital certificates, since it provides no clear structure of how trust is managed in a peer group (whose signatures are trusted and which peers are allowed to sign certificates).

It is interesting to point out the implications of the fact that under the PSE Membership Service peers are authenticated only by being able to access a keystore. The Membership Service is not concerned with the validity (signed by a proper CA and not revoked) of certificate chains. As a result, anybody with access to a private key and a certificate (a self-signed one is sufficient) will be able to join a group using PSE. Those peers which hold the necessary information in order to generate a correct PSE Authenticator, but are not really group members because their certificate is not properly signed are named *interlopers*. These peers can become an annoyance, but are easily dealt with.

As far as the Access Service is concerned, the current JXTA reference implementation offers three kinds of access control: *Always*, *SimpleACL* and *PSE*.

The *Always* Access Service does not really check for access control and allow any operation. It is the default Access Service for peer groups.

The *SimpleACL* Access Service uses Access Control Lists in order to establish which identities may perform the different group operations. The access lists are distributed as parameters within the peer group advertisement.

The *PSE* Access Service provides an interface to PKIX certificate path validation. A trust anchor is set for the validation process and all credentials are

validated against this anchor in order to decide whether the operation is permitted or not.

It is very interesting to point out that all currently provided approaches to the Access Service are strictly tied to peer identity (operations are evaluated according to the claimed identity) and do not check group membership itself. Membership is assumed, even though it may not be always the case, as previously exposed.

5.2 Group membership specification

In order to adapt the model presented in Chapter 4 to the JXTA specification, all its processes must take into account how peers interact with JXTA's Membership and Access Services. In our specification, trust relationships are represented by signed certificates. The only requirements for such certificates are that they are signed by the patron[1] and contain the trustee's public key and identifier, as well as the patron identifier. It is important to remark the difference between a certificate and a credential under our JXTA group membership model. The former represents a trust relationship whereas the latter is a set of information established when a peer joins a group. This information will be used to authenticate to service providers, but is not necessarily a single certificate.

Additional support services are specified when necessary in order to encompass the full trust path creation and evaluation. JXTA supoports different models for such service deployment. In our specification, JXTA's *Peer Resolver Protocol* (PRP) has been chosen, providing an asynchronous query/response model, suitable to a dynamic P2P environment, based on XML formatted messages. Furthermore, PRP allows communication between peers which have not joined the same peer group. This is very important during the registration process, since the prospecting group member is not part of the group yet. This is all the necessary background to undertand the services' specification, so further details for this protocol will not be provided at this stage. Chapter 7 will provide a more thorough explanation.

The summarized steps that a peer B, operating under a JXTA network, must follow in order to access a service under this model are:

1. B searches a patron peer for that group.

[1] See Section 4.1.2 for a detailed definition.

2. B requests to the patron the signature of B's public key (i.e. requests a certificate).

3. If the patron agrees, B obtains a certificate that will be used in the process of proving group membership.

4. (Optional) B is free to repeat steps 1-3 with additional patrons at any later stage.

5. B uses JXTA's Membership Service to establish its identity and credential within the group, by joining it.

6. B accesses a group service through peer A.

7. A asks to B to prove group membership.

8. B looks for a trust path from A to B within the JXTA network.

9. B retrieves the set of certificates that form the trust path (TP) from peers between A and B.

10. B provides TP to A.

11. A evaluates the correctness of TP via JXTA's Access Service. If correct, access based in group membership is granted.

12. (Optional) A new backbone relationship is created between A and B's patron.

13. (Optional) B stores TP, so it may be used in future accesses to A's services.

The different steps in our specification can be divided into two different sets. Steps 1-5, used to generate trust relationships (certificates), are related to interaction with the Membership Service. These steps only happen once during the whole peer's membership lifecycle, just so B becomes a group member (even though steps 1-3 can be repeated to obtain additional patrons). On the other hand, steps 6-11 are related to interaction with the Access Service, where the trust path is validated and the service accessed. The description about how each step is solved in a JXTA network will follow this division.

5.2.1 Membership Service

First of all, it must be noted that, in a JXTA network, whenever a peer wants to join a group, it must previosuly locate the group's *Peer Group Advertisement*[2]. Peer group advertisements are sent by those peers which want to announce the group (usually, at least, the group creator), and must be periodically updated, since it has a set lifetime. Once the advertisement is located, the group may be instantiated and it is possible to interact with group members. However, the peer itself is not considered a group member until it has successfully joined it. Until that moment, it is considered an interloper. The moment such advertisement is no longer available within the network, it is impossible to instantiate a group.

Peer group advertisements contain all the required information and parameters to access any group service, be it group specific or part of JXTA's core (such as the Membership or Access Services). Therefore, any service in our specification may be accessed once the peer group advertisement is retrieved.

Patron search

As part of the definition of the peer group advertisement, additional parameters are appended to the Membership Service field. A list of patron node ID's known by the peer which published the advertisement is included as a service parameter. With this information, prospective members may know where to get a proper certificate in order to access group services (Section 5.2, step 1). As the list of patron nodes changes, new advertisements may take it into account when published. From this list, peers may easily locate available patrons.

An example of Peer Group Advertisement is shown in Listing 1 (Note: ID's have been shortened).

In addition, we define a new group service named *Patron Dicovery Service*. This service queries group members for a list of peer group patrons. Even though the peer group advertisement includes a list of patrons, this service is useful to update the patron list for someone who has already joined the peer group without the need to wait for a new peer group advertisement publication. As a JXTA group service, any group member may run this service. Non-patron peers' response is a list obtained from its own stored patron relationships. Patron peers's response is

[2]JXTA publishes peer resources through advertisements. For a detailed description of JXTA resource publication, the reader can review Section 6.1.3

XML Listing 1 - Peer group advertisement

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jxta:PGA>
<jxta:PGA xml:space="preserve" xmlns:jxta="http://jxta.org">
  <GID>urn:jxta:uuid-2AD...5F02</GID>
  <MSID>urn:jxta:uuid-DEADBEEFDEAFBABAF...000000010306</MSID>
  <Name>SampleGroup</Name>
  <Desc>Collaborative Membership Service Group</Desc>
  <Svc>
    <MCID>urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE0000000505</MCID>
    <Parm type="jxta:PatronList">
      <PeerGroup>urn:jxta:uuid-2AD...5F02</PeerGroup>
      <Patrons>
        <Patron>urn:jxta:uuid-596...4003</Patron>
        <Patron>urn:jxta:uuid-596...1B03</Patron>
      </Patrons>
    </Parm>
  </Svc>
</jxta:PGA>
```

a list obtained from its acknowledged backbone relationships.

The format of the Patron Discovery Service query and response is shown in Listings 2 and 3 (Note: ID's have been shortened).

XML Listing 2 - Patron Discovery query

```
<collab:PatronDiscoveryQuery>
  <PeerGroupID>urn:jxta:uuid-282...8C02</PeerGroupID>
  <Threshold>4</Threshold>
</collab:PatronDiscoveryQuery>
```

XML Listing 3 - Patron Discovery response

```
<collab:PatronDiscoveryResponse>
  <PeerGroupID>urn:jxta:uuid-282...8C02</PeerGroupID>
  <Patrons>
    <Patron>urn:jxta:uuid-596...CB03</Patron>
    <Patron>urn:jxta:uuid-596...DB03</Patron>
    <Patron>urn:jxta:uuid-596...6003</Patron>
    <Patron>urn:jxta:uuid-596...EE03</Patron>
  </Patrons>
</collab:PatronDiscoveryResponse>}
```

The fields used in the messages are the following ones.

- *PeerGroupID* - The unique identifier of the peer group whose patrons are being discovered. This identifier is included in the response in order to correctly process proactive responses, sent without a previous query.

- *Threshold* - A number specifying the maximum number of patrons that should be sent by the responding peer.
- *Patrons* - An element containing a list of patron peer identifiers. The identifiers are relative to the peer group being asked.

It should be noted that the response has the same format as the service parameter included in the peer group advertisement shown XML Listing 1.

Credential generation

Once an existing patron list is retrieved, in order for a prospecting member to become a fully fledged group member, a proper trust relationship must be obtained by getting a certificate signed by a patron peer. This is achieved though the *Sign Service*, allowing peers to request such signatures without the need of out-of-band methods (Section 5.2, steps 2 and 3). This service supports the creation of both patron and backbone relationships. By using this service, it is also possible to obtain certificates from several patrons.

The moment a patron must decide whether to accept or not a query is a very important step. The outcome of such decision is left up to each specific application, since it is impossible to develop a policy that may apply to any scenario. Acceptance will generally be based on the fulfillment of some requirements before joining the group (such as committed resources). An interesting possibility is using trust evaluation schemes [Che01; Bet94]. Another acceptable possibility is just sending the request to the upper level application and relying on direct user input. The user will decide according to some out-of-band information, such as an invitation system.

Registering the signing service in a non-patron peer is pointless, since the certificate will not be a valid one. Even though this behaviour may be potentially disrupting for unsuspecting peers, it is easy to identify which rogue peers are signing requests by comparing patron lists retrieved from different peers, or as soon as a service access is denied.

The details of the Sign Service are shown in the query and response formats in Listings 4 and 5.

The sign query and response parameters are listed here.

XML Listing 4 - Sign Service query

```
<collab:SignQuery>
  <PeerID>urn:jxta:uuid-596...CB03</PeerID>
  <PeerGroupID>urn:jxta:uuid-282...8C02</PeerGroupID>
  <Data type="pkcs\#10">MIIB...PYNvDvHf1</Data>
  <Commendation>...</Commendation>
</collab:SignQuery>
```

XML Listing 5 - Sign Service response

```
<collab:SignResponse>
  <Status>OK</Status>
  <PeerID>urn:jxta:uuid-596...CB03</PeerID>
  <PeerGroupID>urn:jxta:uuid-282...8C02</PeerGroupID>
  <Data type="x509certificate">MIIC...BIBUAMw==</Data>
</collab:SignResponse>
```

- *Status* - Sign request result. The request may have been granted (OK) or denied (DENIED). The responding peer may also respond that the query has been correctly received but it is still to be processed (PENDING). This status is specially useful for applications which rely on direct user input in order to accept or deny a sign request.
- *PeerID* - The unique identifier of the peer who is asking for a signature.
- *PeerGroupID* - The unique identifier of the peer group for which a certificate is being requested.
- *Data* - The request/certificate raw data (Base64 encoded in case of binary data). This element will only exist if a sign request has been accepted (the Status field is OK). It also contains a *type* attribute, which denotes a special identifier specifying the type of request (for a sign request) or certificate (for a sign response). In the case of the sign response, the identifiers chosen for the different types of sign responses are those defined in XMLdsig [W3C02b]. This will allow an easy integration with XML signature. This is a desirable feature since all JXTA protocols are based on XML. In our current specification, PKCS#10 and X.509 certificates are used.
- *Commendation* - Additional optional information that a peer may provide in order to get the sign request accepted. Since the acceptance of a sign request is solved at application level, this field may contain any kind of data. The

specific data format will be defined by each individual application, according to its own needs. The query message just acts as a transparent envelope. Nevertheless, an existing valid trust path may be used as a commendation when creating redundancy and shortening long paths, as explained in chapter 4.

All cryptographic operations, such as signature generation, are arbitrated by the *CryptoManager* interface. Using the CryptoManager enables the integration of peer group operation with any kind of cryptographic module such as hardware cryptographic tokens. This approach also takes into account the fact that not all cryptographic modules are accessed via plain text passwords (for example, using biometrics). Even in cases where passwords are used, sometimes, each provider has its own methods for accessing private keys (for example, a special GUI integrated into the operating system in the the Windows CryptoAPI [Mic07]). Applications will implement the CryptoManager interface according to their own needs and their chosen cryptographic provider. Decoupling cryptographic modules from the Membership Service also allows to manage certificates via out-of-band methods, instead of relying on the application.

Before a certificate is signed via the CryptoManager it must be validated that the presented public key to be signed is actually associated to the trustee's claimed identity. Any peer may invoke the Sign Service in order to get its own public key signed, but under an arbitrary ID. Even though the fact that the response is directly sent to the claimed ID's peer, which means that the attacker will not necessarily receive it, it cannot be considered a completely secure procedure, since the certificate will still be sent through the overlay network. In the case of CA based approaches, it is assumed that the CA acts as TTP, being all powerful and able to completely ensure any peers ID-key binding before generating a certificate. Since a fully distributed system based on peer autonomy is one of our main goals, no common trusted party exists in this proposal's environment.

Key authenticity is achieved in this proposal by using CBIDs, already introduced in Section 2.2.4, which avoids the intervention of a TTP. Other methods exist in order to bind key pairs to identifiers without the need of a TTP, such as the id-based approach or self-certifying keys. However, since JXTA IDs must follow a very specific format, they are not feasible.

A JXTA Peer ID should canonically, uniquely and unambiguously refer to a

peer, being represented in 64 bytes according to the following fields: Peer Group UUID (16 bytes long), Peer UUID (16 bytes long) and identifier type (1 byte long, its value being set at 03 for Peer ID's). The rest of bytes are unused. Notice that according to this format only 16 bytes, the Peer UUID, are really unique to each peer within the same peer group. For that reason, those are the only values which can be manipulated in order to generate CBIDs.

In our approach, CBIDs are generated by applying the SHA-1 [NIS95] hash algorithm to the public key. The result is then considered the Peer UUID in order to construct the full JXTA ID. Since SHA-1 produces 20 bytes but only 16 are needed, the first 4 most significant bytes are discarded. This process is summarized in Figure 5.3.

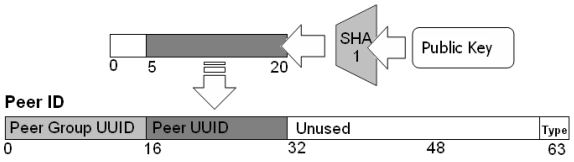


Figure 5.3: *CBID generation*

Using this method for CBID generation, we also provide the system with the capability to omit the need for certificates, permitting the use of any new kind of public key based certificate, not only X.509 Certificates. An additional advantage of this method for generating CBIDs is that since it is based on the SHA-1 hash algorithm, it applies under the UUID type 5 specification [Lea]. This method also allows to integrate peers which use CBIDs with those who do not, since their Peer IDs which are not CBIDs are randomly generated anyway.

It must be taken into account that a single ID is generated from a specific public key, which might be problematic if two different publics keys produce the same UUID field. The probability of two different public keys providing the same UUID can be cast as a collision problem, similar to the Birthday Paradox [Sch96]. The probability of a collision in such scenario is $p(n;d) \approx 1 - e^{-(n(n-1))/2d}$, where n is the number of values drawn from the distribution and $[1,d]$ the distribution range. For a 128 bit UUID, $d = 2^{128}$. According to [Hal05], a typical JXTA peer group has about 32 members before performance starts to degrade. Whith this amount of members, the probability os a collision is about $1/20^{36}$, dismally low.

In order to validate some public key's authenticity via CBID ownership, whenever a Sign Query is received, the contained public key is extracted and then used to generate the source address, *i.e.* the CBID. If the obtained address is the same as the claimed source address, ownership is guaranteed. Otherwise, the receiver may discard the request.

Binding peer IDs with public keys implies that changing ID means changing the public key (and vice versa). This property may be seen as an advantage since it provides an incentive to not continuously change identity, because under certificate-base group membership (such as PSE), a peer changing its public key (or its ID) stops being a member of all groups and must start from scratch. In reputation-based systems, all reputation is lost.

Peer group joining

A peer which has a proper certifcate may finally establish a credential within the peer group context by joining the group using the Membership Service (Section 5.2, step 5). In our Membership Service specification, the join process is also partially managed by the CryptoManager. This interface is passed as the authenticating parameter to the Authenticator and it is used in the application and validation steps to manage credentials, acting as a proxy for any cryptographic module. In contrast, the PSE Membership Service is constrained to a single type of cryptographic provider: a java keystore protected with a password. The location of this keystore is usually the peer's local cache, even though remote access via an URL is also supported. Unfortunately, these methods make it difficult to easily manage certificates via out-of-band methods, since the keystore can only be accessed through method calls to JXTA. Under PSE, it is not possible to use any other kind of cryptographic provider which does not rely on files protected by a plain text password.

The basic CryptoManager methods necessary for peer group joining are:

- *initialize*: Sets up the cryptographic module, in order to be correctly accessed later. Usually, complex providers need some initialization. This includes end-user authentication.

- *check*: Checks whether initialization has been successfully completed and it will be really possible to access stored keys and certificates, returning true or false depending on such initialization. This method also checks whether

a certificate for that group is stored into the CryptoManager. In case none exists, it returns false.

- *getCerts*: Obtains the list of certificates in the cryptographic module. The Authenticator processes this list in order to assemble only those certificates related to the peer group to be joined, generating a peer group credential.

Additional methods for specific characteristics of cryptographic modules may be added (specially regarding on how user authentication is achieved).

A sequence diagram summarizing the full process is shown in Figure 5.4. The full join process is not shown in this figure, but only the part which is particular to our Membership Service specification: from the moment an Authenticator has been made available up to the final step. Once the join method is successfully invoked, a group credential is established. In our approach, the peer cannot even attempt to join the group unless it has previously correctly authenticated to the CryptoManager at the initialize() method.

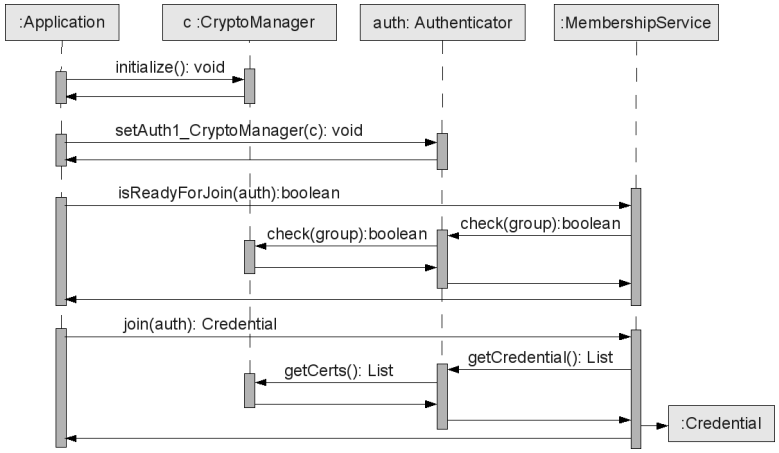


Figure 5.4: *Peer group join process using CryptoManager*

Figure 5.4 can be compared with Figure 5.1 in order to identify the differences between our proposal and the generic Membership Service join process.

Our Membership Service uses a set of certificates as a JXTA peer group credential. Even though open to different types of certificates, currently, only X.509 certificates are used. X.509 certificates have been chosen because they are the

most widely used in cryptographic modules. The peer group ID is stored in the *Organization* (O) certificate field, whereas peer ID is stored in the *Common Name* (CN) fields. The type of relationship (backbone/patron) is stored on the Oraginzational Unit (OU) field. It is worth mentioning that JXTA ID's are very long and cannot be fully encapsulated into a X.509 certificate field. However, only a small part of each ID is specific for any peer or group ID. Thus, only this part, the Peer and Peer Group UUID, respectively for a peer and peer group, is actually stored into each field.

A sample credential is shown in Listing 6 (Note: X.509 certificates have been shortened).

XML Listing 6 - Peer Credential

```
<!DOCTYPE jxta:CollabCred>
<jxta:CollabCred algorithm="SHA1withRSA" type="jxta:CollabCred"
                 xml:space="preserve" xmlns:jxta="http://jxta.org">
  <X509Certificate>MIIC...BIBUAMw==</X509Certificate>
  <X509Certificate>MIID...8dLvZ96==</X509Certificate>
</jxta:CollabCred>
```

After a credential has been established, the CryptoManager is used to operate within the Membership Service as an interface to the cryptographic provider until the peer resigns from the group.

In the case that no proper group credential exists (because steps 1-3 where never properly completed), the isReadyForJoin method always returns false. As a result, it is not possible to join the group. Therefore, our specification does not allow interlopers. A peer must have a credential for that group in order to instantiate it.

5.2.2 Access Service

As exposed in subsection 5.1.2, proof of peer group membership must be required when accessing remote resources. Such proof is the fact that a trust path exists between both peers. However, trust path validation should not be processed during the join process, since group instantiation and the join process in JXTA are locally executed, and thus it would be trivial for a rogue peer to modify the

local code in order to always successfully join the group. Therefore, the Access Service is the one in charge of evaluating a set of certificates forming a trust path.

In our proposal, proper credentials cannot be generated only using local data, they are generated through collaboration with other peers. And, unless a proper credential is somehow generated, no access will be granted at a later stage.

Trust path discovery

The *Trust Path Discovery Service* provides a mechanism to learn whether a trust path between two peers exists, and through which peers this trusted path passes (Section 5.2, step 8). This service correctness requires collaboration between different peers in the group, since the group membership model is based on the principle that each peer is autonomous and only holds data mostly related to itself. It also assumes that no peer maintains memory or state regarding processed queries and the whole process acts in a fully asynchronous way.

In order to retrieve the trust path between two peers, the source peer propagates queries across all its trusted peers. This query contains the final peer ID in the trust path and its own ID as the source peer. In the case that peers receiving the query are not able to retrieve a trust path to the final peer by themselves, they propagate the same query to its own trusted peers, but including its own ID as an intermediate peer. As this query reaches new peers, the query grows to collect the trust path, adding its own ID at the end of the current list, until a peer which knows a trusted path to the final destination is reached. That peer sends the final answer directly to the source peer. A summary of this whole process is depicted in Figure 5.5.

Loops are avoided by comparing the currently retrieved trust path in the query with the list of trusted peers in the processing peer. If a peer already appears in the list, the query is not propagated across that trust relationship.

It has to be pointed out that this service may generate several responses, since several trust paths may exist. Notice also that the trust path need not be retrieved one peer at a time. If a peer already knows a path to the final peer, even though there is no direct trust relationship (for example, because of cached responses to its own queries), it may still respond the query constructing the full trust path. In that case, the peer stops propagating queries.

The Trust Path Discovery query format is detailed in Listing 7. The contents of the response are the same.

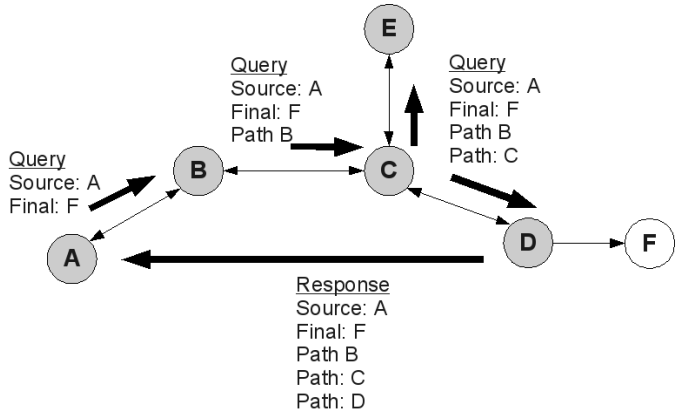


Figure 5.5: *Trust path discovery query propagation*

XML Listing 7 - Trust Path Discovery query/response

```
<collab:TrustPathQuery>
  <SourcePeerID>urn:jxta:uuid-596...CB03</SourcePeerID>
  <FinalPeerID>urn:jxta:uuid-596...6003</FinalPeerID>
  <PathPeerID>
    <Position>1</Position>
    <PeerID>urn:jxta:uuid-596...DB03</PeerID>
  </PathPeerID>
</collab:TrustPathQuery>
```

The query and response parameters follow:

- *SourcePeerID* - Identifier of the starting peer in the trust path.
- *FinalPeerID* - Identifier of the final peer in the trust path.
- *PathPeerID* - Identifier of an intermediate peer in the trust path. These fields are ordered according to transit from the source peer to the final peer.

Since each peer manages its own list of trusted peers, the Trust Path Discovery Service also allows to check revocation status, since no queries will be propagated through revoked trust memberships. This is in contrast with other membership services, such as PSE, where revocation does not exist and it is only possible to voluntarily resign from a group.

Trust path generation

The *Certificate Retrieval Service* can be used to generate trust paths by retrieving certificates from other peers (Section 5.2, step 9). Once a trust path is located, all individual certifcates may be retrieved using this service in order to proceed to its validation. Several retrieval petitions may be encapsulated in a single request, in order to improve efficiency. Again, responses may be sent in a proactive way without the need of a previous query. Additionally, this service allows asking for certificate revocation status (useful for Section 5.2, step 11). All requests are only relative to the information stored within a peer.

The Certificate Retrieval Service query and response format are shown in Listings 8 and 9.

XML Listing 8 - Credential Status query

```
<collab:CertRetrievalQuery>
  <PatronPeerID>urn:jxta:uuid-596...CB03</PatronPeerID>
  <TrustedPeerID>urn:jxta:uuid-596...6003</TrustedPeerID>
</collab:CertRetrievalQuery>
```

XML Listing 9 - Credential Status response

```
<collab:CertRetrievalResponse>
  <TrustedPeerID status="OK">urn:jxta:uuid-596...6003</TrustedPeerID>
  <Data type="x509certificate">MIIC...BIBUAMw==</Data>
</collab:CertRetrievalResponse>
```

The fields in the certificate status query are:

- *PatronPeerID* - The queried certificate's patron peer ID. This field may appear multiple times, containing different ID's.
- *TrustedPeerID* - The queried certificate's peer ID.

For the certificate status response, the fields are:

- *TrustedPeerID status*- Response tied to a TrustedPeerID request field. Credential status for a peer with a specific ID. The status attribute may be OK, REVOKED or UNKNOWN. The latter case may occur if the query asks for the status of a certificate which was not generated by the responding peer.

- *Type* and *Data* - These fields are a response to a PatronPeerID request field. They accomplish the same function as the ones used in the Sign Service. The ID from the original request may be obtained form the certificate content.

Trust path evaluation

Each time a service is accessed, the received trust path (Section 5.2, step 10) is checked by the Access service. Since the group membership and access control model is based on collaboration between peers, the Access Service must operate in a fully asynchronous way in order to take into account the nature of ad hoc environments. Some delay is to be expected between access to the service being asked and a final decision regarding access control being reached. For that reason, in our specification the doAccessCheck operation will return the result "UNDEFINED" until it is ready to provide a definite answer. The final behaviour is similar to that of the isReadyForJoin in the join process (see Section 5.1.1), which must be periodically polled in order to know when it is possible to proceed.

The basic trust path evaluation (Section 5.2, step 11) relies on validating the signature on each certificate and checking whether a trust path actually exists from the service provider to the service client, by chaining each certificate through issuer and subject identifiers. For example, if we define $Cert_x^y$ as a certificate where x is the subject and y is the issuer (the signer), the following trust path is correct $\{Cert_A^B, Cert_B^C, Cert_C^D, Cert_D^E\}$. Thus, A would be able to prove group membership to E when accessing any of E's services. Credential revocation status may be optionally checked by using the Credential Status Service.

Key identity was ensured at the certificate generation step, since in a trust path the validating peer acts as the root turst anchor. It is assumed that a patron does not sign any certificate unless the public key is deemed properly authentic. If that is not the case, accepting the result is under the validating peer's own responsibility, since the authenticated peer is accessing his services.

5.3 Chapter Summary

This chapter has specificed the peer group membership and access control model defined in Chapter 4 using the JXTA framework, with certificates as a means to represent trust relationships. This specification integrates with the core services

provided by JXTA in order to control peer group membership: the Membership and Access Services. Group management is achieved via additional services that take into account the idiosyncrasies of JXTA's messaging capabilities: the Patron Discovery, Sign, Trust Path Discovery and Certificate Retrieval services. In addition, key authenticity is provided in a lightweight manner with CBIDs, which avoids relying on a TTP.

The specification may be adapted to different cryptographic modules through the CryptoManager interface, in order to achieve key management with different certificate types. The proposed services also provide the fundamentals for deploying the unlinkability scheme exposed in Chapter 4 without the need of additional protocols. Unfortunately, since, at this stage, JXTA is not capable of providing an underlying anonymous transport, anonymous authentication cannot be included into the Membership Service.

Part II

Securing peer group operations

Chapter 6

A Survey on JXTA's security

In this chapter, a survey of the current state of security in JXTA for basic peer operations is provided [AM08b; AMss]. This is a very important step in order to assess how to efficiently deliver peer group operation security under JXTA, since it provides a clear idea of which are the main issues with the current specification.

A lot of research efforts in the field of P2P have mainly focused towards strictly functionality issues such as scalability [Dai09], efficient message propagation across the network [Wan07] or access to distributed resources [Zer04]. At present time, the maturity of the P2P research field has pushed through new problems such as those related with security. For that reason, security starts to become one of the key issues when evaluating a P2P system. It is important to determine which security mechanisms are available, and how they fit to every specific scenario. Even at the cost of some impact on performance, a security baseline must be kept during operation in any P2P system, in order to ensure some degree of correctness even when some system components will not act properly.

Comprehensive reviews regarding security issues in P2P systems can be found in [Wal03; Vro06]. Systems are compared from two different standpoints. On one hand, P2P security is examined based on system capabilities, such as routing or data storage, analyzing the security threats they imply. On the other hand, another approach is to perform a security analysis based on individual security goals, such as reputation, availability or access control. Some attempts to provide metrics to measure the security degree of a P2P system have also been proposed.

For instance, in [Sit02] a security framework for such purpose is proposed, but only on regards to hash-table based resource location and distribution. These generic P2P system security reviews analyze system capabilities or security goals in an isolated manner and for that reason, a more accurate security analysis, for instance based on the peer's life cycle, cannot be performed.

Regarding the JXTA P2P middleware, although the platform has been available for several years, studies have been mainly concerned with performance [Hal05; Dai06; Ant05] but not many have discussed such a sensitive issue as the development of secure applications. The most complete effort to present the available security mechanisms in JXTA can be found in [Yea02]. However, it can be mainly considered as a statement of design goal achievements. It does not provide a thorough review of all available security mechanisms or a clear idea of how they really work, making it difficult to assess in detail which are the real vulnerabilities in JXTA. On the other hand, JXTA documentation on regards to security is scarce and, in many cases, source code has to be directly reviewed in order to understand how its security mechanisms really work. Consequently, the results of this research work will benefit security-aware platform developers and designers which want to create JXTA applications, by providing them an up-to-date detailed list of which security related issues should be taken into account. JXTA users may also benefit by realizing which may be the limitations of their applications on the scope of security, so they may take additional measures in order to guarantee a completely secure environment.

The chapter is organized as follows. In Section 6.1, an overview of the JXTA platform is provided in order to understand its main characteristics and methods of operation. Even though some specifics of group membership where already presented in Chapter 5, in this chapter the whole framework will be presented. Following, in Section 6.2, the basic evaluation model is described by categorizing basic peer operations and threats under the JXTA model. Section 6.3 presents the security analysis. This analysis includes both pure JXTA security mechanisms and additional existing security improvement proposals that are not currently integrated into the base JXTA framework. The final conclusions for this chapter are exposed in Section 6.4.

6.1 An overview of JXTA

This section provides a general overview of the main JXTA concepts and components in order to understand the peer operations explained in Section 6.2 and their security concerns. A detailed description of JXTA can be found in [jxt07; SUN07].

The fundamental JXTA architecture is divided into three distinct layers, as shown in Figure 6.1.

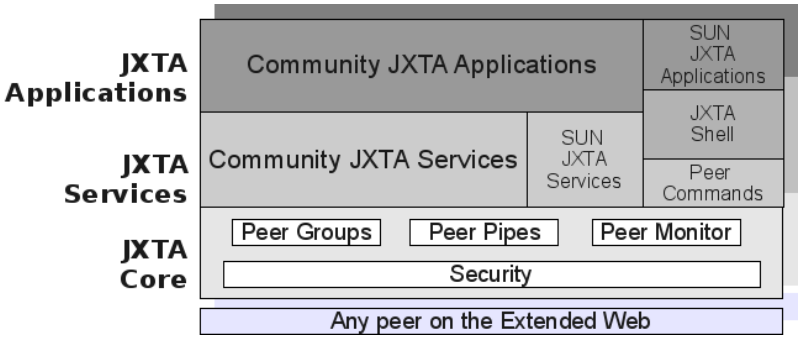


Figure 6.1: *JXTA's layered architecture*

The *Core* layer contains the minimum and essential characteristics, common to all P2P networks. Such operations include peer creation, discovery and communication, even when behind firewalls or Network Addres Translation (NAT), as well as basic security services.

The *Services* layer includes all network services which are not absolutely necessary, but provide desirable capabilities such as resource search, indexing, storage and sharing.

The top layer is the *Applications* one, which includes any application deployed using the JXTA framework, such as instant messaging, file sharing or content management.

Notice that the distinction between services and final applications may not always be clear, since what a client may consider an application may be considered a service by another peer. For that reason, the system is designed in a modular way, letting developers choose the set of services and applications which most satisfy their needs. All JXTA components are within these three layers.

6.1.1 Peers

Every peer in the JXTA virtual network is identified by a unique Peer ID, operating in an independent and asynchronous manner on regards to other peers. However, some dependencies may exist depending on which roles they partake.

Usually, peers act as *edge peers*, which could be considered the standard peer type in any desktop application on most devices. They implement the JXTA Core and Services layers as shown in Figure 6.1 and may interact with any JXTA protocol. Edge peers may also partake two additional roles in order to avoid some specific network constraints: *minimal* and *proxy*. This decision usually depends on its hardware or bandwidth capabilities.

Devices with specific resource constraints (memory, CPU) may act as *minimal peers*, which only implement the JXTA Core layer. They are usually simple devices such as sensors or domotics. In order to use any necessary service to operate within the network, they must rely on *proxy peers*. A proxy peer summarizes and answers requests on their behalf.

A very important role is that of *rendezvous peers*, which maintain a global index of available resources and help other peers find network services. They also act as beacons which newly connected peers may use in order to join the network. For that reason, rendezvous peers are usually well-known ones, with a DNS name or a static IP address.

Connectivity between peers physically separated from the JXTA network, because of firewalls or NAT, is achieved by means of *relay peers*. They provide the ability to store and resend messages for unreachable peers and, by exchanging route information, message transport across several relay peers is possible in a transparent manner. Nevertheless, peers always attempt direct connections before using a relay.

6.1.2 Protocols

As explained, JXTA defines a set of protocols (six, specifically) which enable the deployment of P2P networks. Using these protocols, peers may collaborate in a fully autonomous manner by publishing and discovering available resources within the network. Peers may also cooperate in order to route messages, allowing full communication, without the need for them to understand or manage different network topologies.

All JXTA protocols are asynchronous and based on a request/response model, which means that for any given request, zero, one or many responses may be received. A brief description of each protocol is given:

- The *Peer Discovery Protocol (PDP)* allows peers to publish their own resources and make them available to other peers. Any kind of peer may send PDP messages. This protocol is the default discovery protocol, but it is possible for applications to implement and deploy their own protocols.
- In order to obtain information about other peers, the *Peer Information Protocol (PIP)* is used. Using this protocol, it is possible to query peer capabilities or monitor its behavior.
- Peers use the *Peer Resolver Protocol (PRP)* in order to send requests to one or several peers and manage responses. The PRP protocol uses an unique ID associated to every request, which is included in messages. Other core protocols, such as PDP, make use of PRP in order to create its own requests.
- The *Pipe Binding Protocol (PBP)* establishes virtual communication channels between peers named *pipes*, acting as abstract endpoints above any physical network.
- Routes between peers are found with help of the *Endpoint Routing Protocol (ERP)*. Whenever some peer is about to send a message to a destination but does not know any path, an ERP message is sent to other peers asking whether they know a path.
- Finally, the *Rendezvous Protocol (RVP)* is responsible for the efficient propagation of messages within a group of peers, allowing peers to connect to services and exchange messages.

6.1.3 Resource publication

Any kind of resource available within the JXTA network, including peers, peer groups, pipes or services, is described by an *advertisement*.

Advertisements are XML documents containing an unique ID and all information regarding that resource and how it may be accessed and exchanged between peers using the JXTA protocols. Peers cannot access a resource without previously retrieving its associated advertisement. Every peer maintains a local cache

where all received advertisements are stored for a later use. The local cache directly makes use of the peer's file system in order to organize its content via directories and files.

A sample advertisement is shown in XML Listing 10.

XML Listing 10 - Peer Advertisement

```
<xs:element name="PA" type="jxta:PA"/>

<xs:complexType name="PA">
  <xs:sequence>
    <xs:element name="PID" type="jxta:JXTAID"/>
    <xs:element name="GID" type="jxta:JXTAID"/>
    <xs:element name="Name" type="xs:string" minOccurs="0"/>
    <xs:element name="Desc" type="xs:anyType" minOccurs="0"/>
    <xs:element name="Svc" type="jxta:serviceParams"
                minOccurs="0" maxOccurs="unbounded"/>
  <xs:sequence>
</xs:complexType>
```

Any peer may publish an advertisement to announce the availability of a particular resource by using two different methods: local and remote publication.

In *local publication*, the advertisement is indexed and stored the peer's local cache. Following, the advertisement's index is pushed to a rendezvous peer and is then distributed and replicated between all rendezvous in the global super-network peers, using the Shared-Resource Distributed Index (SRDI) service [Tra03b; Tra03a]. The rendezvous network acts as a remote index cache.

By using this method, it is possible for peers outside the local network (out of broadcast range) to retrieve group advertisements by asking a rendezvous peer. It also enables peers which were off-line for some time to retrieve advertisements published during its disconnection. Whenever a peer receives the advertisement, the latter is indexed, stored into the local cache and assigned an expiration date.

The advertisement retrieval mechanism is outlined in Figure 6.2.

It must be remarked that during the publication process, the original advertisement is always kept in the peers' local cache, only its index is distributed. This means that in case the peer goes offline, the advertisement will become unavailable. That makes sense, since also the resource the advertisement publicizes will be unreachable.

In *remote publication* not only indexes are distributed, but the full advertisement itself via the JXTA propagation mechanism. This method is useful in case that the advertisement must be reachable even when the publishing peer is

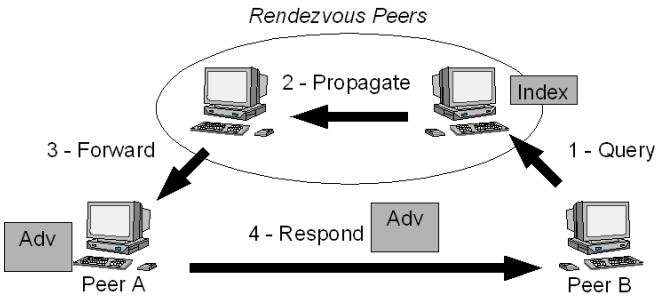


Figure 6.2: *Advertisement retrieval from Rendezvous peers*

offline. However, under the remote publication method, no assumptions can be made about which peers will really store the advertisement and for how long.

In both methods, when the expiration date is reached, the advertisement is considered stagnant and flushed from the cache, unless the same advertisement is received again, which renews its expiration date. Advertisements may be periodically retransmitted in order to attain permanency or update parameter changes.

As can be seen, advertisement publication and discovery are very important steps in JXTA's peer operation.

6.1.4 Messaging

JXTA peers use *pipes* in order to exchange messages and access available services. JXTA messages are XML documents with ordered message elements which may contain any type of payload. Messages are the basic data exchange unit in JXTA and all protocols are defined as a set of messages exchanged between peers.

Pipes provide an asynchronous, unidirectional and unreliable communication channel by default. However, bidirectional reliable channels may be provided on top of them. They offer two operation methods: *unicast* pipes, which allow one-to-one communications, and *propagation* pipes, which allow one-to-many.

JXTA pipes are an abstraction and are not bound to a specific IP address or port. They have a unique ID and are published just like any resource in the JXTA network, so any peer may update them whenever its physical location changes. Both input pipes, used for message reception, and output pipes, using for message sending, are considered *pipe endpoints*, an actual destination in the physical

network. Endpoints are dynamically bound to peers via the PBP protocol.

6.1.5 Peer groups

JXTA introduces the concept of *peer group*: a collection of peers with a common set of interests. This concept is one of the main foundations of the JXTA architecture and is prevalent throughout all its specification [SUN07]. Offering the possibility to create different (but not necessarily disjoint) groups of peers operating under the same overlay network allows to segment the network and offers a context for peers to publish and access different services.

Peer group boundaries provide a secure framework in order to grant or deny access to the offered services. Peer groups form logical regions whose boundaries limit access to group resources, in a way similar to a VPN [Fer98], operating at the application layer. Other interesting uses are the ability to provide a scoping or monitoring environment, where different classes of traffic and advertisements are limited to only peer group members.

Peer groups are published, discovered and accessed just like any other resource in the network, by means of advertisements.

In order to allow peer group management, JXTA defines the basic primitives for group membership and access control: the Membership and the Access Service. Both are core services which make use of the base JXTA protocols specification in order to achieve their ends. However, JXTA only defines the primitives, while specific applications may implement their own Membership and Access Services depending on their needs (see Section 5.1 for a detailed description of both services).

The Membership Service manages identities within a peer group, providing each group member a *credential*. Peers may include this credential in messages exchanged within a peer group in order to allow other members to know who is making a request. With this information, the JXTA Access Service may evaluate the credential when a service is accessed and decide whether the request will be granted or denied.

A peer establishes its credential within a peer group by successfully *joining* it. Before a peer may join a group, it must be authenticated by providing a correct *Authenticator* to the Membership Service. An Authenticator contains all the required information in order for the Membership Service to check that some

requested identity can be granted. Each different Membership Service specification provides its own definition of an Authenticator, suited to its needs and inner workings.

The Access Service provides a single primitive in order to check a credential for a privileged operation. The possible results are disallowed (access denied), permitted (access granted), permitted but expired (the operation would be permitted but the credential has expired) and undetermined (unrecognized credential).

6.2 Security evaluation model

The first step in order to assess the security degree of JXTA applications is to identify which is the standard peer lifecycle, so that it becomes clear which are the most common operations and, consequently, which deserve better attention on regards to security. Once such operations have been identified, a list of usual security concerns in P2P environments is provided. Our security evaluation model will be based in the cross-reference of standard operations and such threats, in order to evaluate how the system is protected against each one.

6.2.1 Standard peer operation cycle

This section describes the standard peer operations for a peer participating in a JXTA network. The order in which they are presented is a logical one for most scenarios. However, it must be taken into account that such order may vary depending on the peer's role.

Platform startup

This is the initial step in order to setup the platform in the physical device which will hold a JXTA peer. This process mainly consists in loading the required libraries and creating the necessary data structures for network connectivity.

At startup, all peers automatically join a default bootstrap peer group named *netPeerGroup*. This peer group is well known to all peers, since it's ID is hard coded into the platform distribution. Peers may decide to stay only in this group or join others once they are connected to the JXTA network. At this stage, edge peers may also try to reach relay or rendezvous peers depending on their local configuration.

Peer group joining

A peer will join those peer groups formed by peers it wants to interact with. This step usually follows startup, so all later operations are only within the boundaries of those specific peer groups and not the whole network. The peer locates the peer group advertisement and joins it via the peer group's Membership Service. In the case that such peer joins the group for the first time, it must locate the peer group advertisement via PDP. From that point onward, the advertisement is already stored in its local cache.

A peer may join several groups, which means that this operation may be performed several times.

Resource publication

Any resource that peer holds and wants to make available to the rest of peer group members is announced by creating and publishing an advertisement as described in Section 6.1.3. This step includes announcing its own presence, by publishing a peer advertisement, or creating a new group, by publishing a peer group advertisement.

Resource discovery

Available resources in a peer group are discovered by retrieving their advertisements via the PDP protocol. This includes discovering other peers and pipes in order to initiate message exchanges. Usually, pipe advertisements are embedded into other more generic advertisements such as service advertisements.

Message exchange

This would be the most frequent operation during any peer operation cycle. Message exchanges may occur at core protocol level or at service access. At core protocol level, JXTA core protocols are directly used. At service access, two pipe endpoints are established between the communicating peers. An outbound pipe is created by the peer which acts as a client, in order to send requests, and an inbound pipe is created by the peer which acts as a server, in order to receive and process requests.

Platform startup	Load platform Join *netPeerGroup* Open network listeners Open local cache
Peer group joining	Retrieve group advertisement Instantiate group Fill in Authenticator Join
Resource publication	Create advertisement Local publication Remote publication
Resource discovery	Locate advertisement Store advertisement in local cache
Message exchange	Open pipe Send messages Receive messages Access Service check
Disconnection	Close connections Shutdown platform

Table 6.1: *Basic operation substeps*

Disconnection

The peer resigns from all peer groups and goes to offline state in a tidy manner.

This list of operations takes into account some degree of abstraction, since each one actually represents a set of more basic steps. In Table 6.1, a breakdown of each operation into such basic steps is provided. A more detailed explanation can be found in [SUN07]. Nevertheless, from a security assessment standpoint, the chosen degree of abstraction is enough to provide a clear idea of which are the possible scenarios during any peer's full operation cycle, from startup to disconnection.

6.2.2 Security threats in P2P networks

The standard security threats in the traditional client/server environment are still valid in P2P environments. Furthermore, the P2P paradigm shift introduces new concerns that must be taken into consideration when designing P2P frameworks. The move from a passive stance (client) to an active one (peer) in the network easily propagates such concern across all its members. Security attacks in P2P

systems are classified into two broad categories: passive and active [Gov02].

Passive attacks are those in which the attacker just monitors activity and maintains an inert state. The most significant passive attacks are:

- *Eavesdropping (Evs)*, which involves capturing and storing all traffic between some set of peers searching for some sensitive information (such as personal data or passwords).
- *Traffic analysis (TAn)*, where the attacker not only captures data but tries to obtain more information by analyzing its behavior and looking for patterns, even when its content remains unknown.

In active attacks, communications are disrupted by the deletion, modification or insertion of data. The most common attacks of this kind are:

- *Spoofing (Spf)*, in which one peer impersonates another, or some outside attacker transforms communications data in order to simulate such an outcome.
- *Man-in-the-middle (MitM)*, where the attacker intercepts communications between two parties, relaying messages in such a manner that both of them still believe they are directly communicating. This category includes data alteration between endpoints.
- *Playback or replay (Pb)*, in which some data exchange between two legitimate peers is intercepted by the attacker in order to reuse the exact data at a later time and make it look like a real exchange. Even if message content is encrypted, such attacks can succeed so long as duplicate communications are allowed and the attacker can deduce the effect of such a repeat.
- *Local data alteration (LDA)*, which goes beyond the assumption that attacks may only come from the network and supposes that the attacker has local access to the peer, where he can try to modify the local data in order to subvert it in some malicious way.

Apart from security threats that take into account a malicious attacker, it is also very important to take into account *Software Security Flaws (SSF)* in a security survey. Specifically, which steps are taken by the developers in order to minimize the probability that a bug that may later jeopardize a deployed system.

6.3 Security evaluation

From the standpoint of basic security requirements which are desirable in JXTA, they are very similar to those of any computer system: confidentiality, integrity and availability. In order to achieve them, these requirements should translate into an architecture that includes authentication, access control, audit, encryption, secure communication and non-repudiation.

JXTA remains neutral to cryptographic providers or security schemes. In its initial conception it does not mandate any specific security solution, providing a generic framework where different approaches can be plugged in. Enough hooks and place holders are provided in order for each specific application to implement its own security solution. Nevertheless, in a present day P2P framework, relying in the fact that each application will build from scratch its own security solution is not enough, since it will usually mean that security will be overlooked, as it is often the case. It is very important that basic tools and functionalities are already available, providing a default degree of security but allowing further modularization if necessary. As such, basic security services (encryption, integrity and authenticity) should be provided, at least, at the Core layer, even though some applications may chose not to use them.

In this section we will analyze whether the current iteration of JXTA (version 2.5) is up to this desiderata for each of peer basic operation and according to the standard threats to P2P networks.

6.3.1 Platform startup

During the framework startup phase, since the networking capabilities are not operative yet, no threat related to a networked environment applies. However, it does make sense to take into consideration local data alteration on such libraries, since it is at this precise moment that JXTA libraries are loaded into the system.

Due to the open nature of JXTA, the official project page protects software integrity with SHA1 digests [NIS95], in order to avoid malicious distributions. For that reason, a local attack is necessary to actually deploy fake libraries into the system. The current Java distribution also uses code signing to provide a basis for integrity and authenticity checking of installed libraries. Specifically, it makes use of the Java jarsigner [SUNa] tool when the source code is built. The necessary keystore information to sign the code, both the private key and certificate, is

distributed with the source code. Unfortunately, this approach allows anybody to sign malicious the code, since it uses a self-signed certificate, and the keystore password is easily available in the build files (it is *jxta.platform*), meaning that the whole keystore content may be easily compromised. This is unavoidable if it must be possible for anybody to build the libraries. The only solution would be to have an official distribution, built by a trusted party, whose keystore information is not widely available.

Apart from data alteration, it is worth analyzing JXTA's libraries security from a Software Security Flaw standpoint. JXTA is an Open Source Software (OSS) project, which is a good indicator when specifically analyzing security [Cal01]. Obviously, security through obscurity should not be applied, so any software security mechanism which depends upon secrecy tends to eventually fail, as bugs or security flaws are discovered. Since JXTA code is public [SUN01], it has been audited by a large number of individuals all across the Internet. The use of an OSS approach not only ensures current security, but allows direct improvement from the JXTA developer community, maximizing the networking effect.

Nevertheless, it is worth mention that opening the source code creates the opportunity for individuals to review security, but it cannot guarantee that such reviews will occur. There is also the fact that no guarantee can be made that a review will find any particular security flaw in a system, but that problem is also common to closed source projects. In any case, OSS allows developers with security concerns to directly assess whether the JXTA framework is up to their needs.

6.3.2 Peer group joining

The first step in order to join any peer group is to retrieve its peer group advertisement. The implications of this specific substep will be explained in the next subsection. Therefore, we will focus here on the actual group instantiation and join operation.

As described in subsection 6.1.5, the Membership Service is the key security mechanism for the group join operation. Through this service, peers claim identities by proving they are the legitimate owner. This service is defined as generic in the JXTA specification, so each application may implement it according to its own needs. However, the JXTA reference implementation, as far as version 2.5

[jxt07], provides three available Membership Services which are ready to use.

The *None* Membership Service is intended for peer groups which need no authentication. Since any peer may claim any identity, it is recommended that credentials should only be used within the group for purely informational purposes. This service is widely used in applications with no security concerns.

The *Passwd* Membership Service relies on a Unix-like username and password pair for peer authentication. In order to claim an identity, the correct password must be provided. The list of pairs (username and password) is distributed to all group members, which means that the password file equivalent freely roams across the overlay network, which makes this method completely insecure. In fact, this group membership service was created as a sample and a means of testing and it is advised in the JXTA documentation that it should never be used in any serious application.

The default Membership Service is *PSE*, which stands for *Personal Security Environment*. This service is the only one that is considered secure and the one that will be analyzed.

PSE provides credentials based on PKIX certificates. As shown in XML Listing 11, any number of such certificates may be included as *Certificate* elements in the PSE credential, together with the Peer Group ID and the subject's Peer ID. The credential itself is also signed.

XML Listing 11 - PSE Credential XML schema

```
<xs:complexType name="jxta:PSECred">
  <xs:sequence>
    <xs:element name="PeerGroupID" type="jxta:JXTAID"/>
    <xs:element name="PeerID" type="jxta:JXTAID"/>
    <xs:element name="Certificate" type="base64binary"
             minOccurs="1" maxOccurs="unbounded"/>
    <xs:element name="Signature" type="base64binary"/>
  </xs:sequence>
</xs:complexType>
```

The authentication procedure in order to join a PSE peer group can be summarized as follows:

1. The user introduces its personal password.

2. The peer initializes an Authenticator for the peer group the user wants to join, using the provided password and the peer's ID.

3. Using this information, an encrypted keystore in the local cache is located and opened, or created if not already existing.

4. The user's certificate chain is retrieved.

5. Such certificate chain becomes the group credential for that peer.

6. The peer may interact with other ones in the same peer group.

7. The private key in the keystore is used by the peer in secure protocols when needed.

All the enumerated steps in the join process using the PSE Membership Service are completed via local calls to JXTA libraries. For that reason, peer group joining is not concerned with network-based attacks (eavesdropping, traffic analysis, MitM or replay), since there are no real network based operations. It also means that any vulnerability in the join process must be exploited by a local attacker.

As we can see, three actors interact in this process: the final user (the human being in front of the computer screen, or some agent), the peer (the application) and the peer group (JXTA libraries which control group access). That means that two spoofing methods must be taken into consideration:

- Impersonating the user: Unauthorized access to keystore content. This is equivalent to taking control of another user's peer.

- Impersonating the peer: Unauthorized identity claim and credential generation. This is equivalent to being able to claim any identity within a peer group.

In the first case, all security relies on the strength of keystore encryption and its password. Unfortunately, the keystore is stored as a simple file, which may be easily copied and distributed, and no mechanisms exist in order to plug in more advanced methods of key management (such as cryptographic hardware tokens).

Regarding peer impersonation, it must be pointed out that peers under a PSE Membership Service are authenticated only by being able to access a local keystore. During the process, the Membership Service is not concerned with the

validity of certificate chains (whether it is signed by proper Certification Authorities or not expired), and the certificate content is never checked. Credentials are actually checked during the message exchange operations, as will be explained in Section 6.3.4.

As a result, anybody with access to a private key and a certificate (a self-signed one would be enough) will be able to correctly join any group using PSE and initially claim any identity. For that reason, the default join scenario is easily threatened by peer spoofing attacks, since anybody may create a valid keystore using public domain tools [SUNb]. There is no real security on peer identity claims on regards to the Membership Service. These kind of peers which hold the necessary information in order to generate a correct PSE Peer Group Authenticator, but are not really group members because their certificate is not properly signed are named *interlopers*. In this scenario, they can become an annoyance, but can be spotted and dealt with when accessing services.

A further concern that developers should take into account when using the PSE Membership Service is the fact that no revocation mechanism is provided. JXTA takes into account the possibility that a group member voluntarily resigns from a peer group, but does not provide the capability to expel a group member for some reason such as malicious behavior. No primitives exist within the framework which may somehow allow this. Developers must create their own revocation schemes from scratch.

To sum up, we can see that PSE is a kind of toolbox that allows the implementation of different models based on securing identities via digital certificates, but it does not provide a clear structure of how trust is managed in a peer group (whose signatures are trusted or which peers are allowed to sign certificates). To properly configure PSE, the application must define the real trust anchors and some method that guarantees key authenticity. Under this scenario, trust anchors may take the form of PGP trust chains [Gar94], where every peer may issue certificates, or a single CA. In the latter case, it may based on a fully centralized approach, where a single peer holds the CA private key, or a distributed one [Des89], where the private key is split between several peers, which must collaborate to issue certificates.

In scenarios where each peer may generate certificates, PSE may not properly, since it is not possible to easily guarantee key authenticity with the provided primitives. This is a weakness developers of JXTA agree that should be addressed

[Yea03]. Even under this assumption, it must be remarked that correctly setting a trust anchor in a pure P2P environment is not an easy task. First of all, in-band certificate generation procedures are easy prey to MitM attacks. In addition, when peer equality must be preserved, the proposed solution should avoid that the peer group may become entirely reliant on some peer, which gets some extra power above the rest.

One of the original creators of JXTA, Yeager, provides a specific trust model for the membership service in [Yea03], which tries to solve the problems previously described. Without actually recognizing a specific CA for each peer group, he proposes that rendezvous peers become the system's trust anchors, the main reason being that they are well protected. Each edge peer uses rendezvous certificates as root certificates in order to ensure key authenticity. Furthermore, to acquire a certificate the peer must be authorized via an LDAP (Lightweight Directory Access Protocol) [Wah97] directory with a recognized protected password. Rendezvous peers may use a secure connection to the LDAP service to authorize requesting peers. The rendezvous peer's root certificate is securely distributed out-of-band by being directly included into the JXTA libraries.

However, developers should take into account that this proposal is ultimately based in a centralized approach and peer groups become heavily reliant on external entities, which makes the system unfeasible in ad hoc environments. Furthermore, the proposal does not specify who configures or manages the LDAP service. It is assumed that some group administrator will do it, which makes it a logical approach in an enterprise environment, but moves away from a pure P2P model.

Non-core JXTA group membership

Due to its open project approach, the JXTA platform offers additional security mechanisms, not currently included in the standard distribution, that have been proposed in order to tackle some of the described issues, by providing additional Membership Service implementations.

An initial proposal can be found in [Li03], but its similarity with the Passwd Membership Service inherits most of its pros and cons. For that reason, it cannot be considered completely secure either.

Another proposal [Kaw04] is similar to the trust model proposed by Yeager, adding extra capabilities upon the basic Membership Service. This approach is based on a centralized PKI and a basic challenge-response protocol [cha96]

as a means for authentication during the join process. Its main contribution is to provide a method which peers may use in order to authenticate the group itself. It also allows to check peer group membership during the join process and defines a trust anchor based on both an external centralized LDAP server and a sign server. Using this method, it is no longer possible to easily spoof peer identities. The LDAP server provides peer certificate management and the sign server deals with group authentication, keeping a secure copy of a private key which represents the whole group. Again, developers should take into account that this proposal becomes reliant on external entities and cannot be considered a pure P2P approach.

More elaborated proposals are presented in [Yun05; Amo05], based on joint authorization by multiple peers under voting schemes in order to maximize decentralization. Under this approach, JXTA credentials are signed certificates issued by a group CA, however group access is based on an agreement reached between several group members, instead of being entirely up to the CA's decision. The main difference between both proposals is that [Amo05] includes a rank system, where peers who join the group ("newbies") have the least privileges, but they may rise to higher positions as they contribute to the group.

Finally, a proposal which moves away from a centralized PKI and uses a web-of-trust approach is presented in Chapter 5.

6.3.3 Resource discovery and publication

Resources provided by peers in the JXTA network are represented by XML documents named advertisements, as detailed in subsection 6.1.3. In order to access such resources, their advertisement must be somehow retrieved. Advertisement discovery and retrieval is achieved via message exchange using the PDP and PRP core protocols (see subsection 6.1.2). Since it is a network-borne operation, we can focus on all threat types. We will discuss both resource discovery and publication in this section, since both share the same security mechanisms (only data flow direction changes between both operations)

In the current JXTA reference implementation, advertisements may be secured at two different levels: at transport layer and at advertisement layer. Since securing at transport layer means considering an advertisement as a standard message, the provided security methods will be explained in subsection 6.3.4, where security

in messaging is analyzed. In this subsection we will focus on advertisement-specific methods.

At advertisement layer, security is achieved by digitally signing advertisements at application level by using a special type of advertisement named *Signed Advertisement*. The direct use of digital signature on the advertisement makes it possible to support both local and remote publication (via propagation to multiple peers). As a precondition to use this special type of advertisement, it is mandatory to join a peer group that uses the PSE Membership Service, since the necessary cryptographic keys to generate and validate the signature are obtained from the keystore and credential associated to this service. Signed advertisements will only be sent to members of that group. In any peer group which does not use this service, signed messages cannot be exchanged between its members. As a result, any security concern related to PSE is inherited by signed advertisements, such as the lack of real authenticity without setting a trust anchor at application level.

The XML schema definition for a Signed Advertisement is shown in XML Listing 12. It contains the signer's credential (the `PSECred` element, a credential for a PSE Membership Service), the signature and the original advertisement. The `Advertisement` element encapsulates the original XML advertisement as plain text encoded via the Base64 algorithm [Jos03]. The content of the `Signature` element is generated by applying the RSAwithSHA1 algorithm to the original advertisement, XML formatted (not its Base64 encoded form). In order to feed the algorithm, the XML data is processed as plain text. The result is henceforth Base64 encoded in order to be represented as plain text into the XML document. Once a signed advertisement is received by a peer, the signature is validated, the actual advertisement extracted and then stored into the local cache.

XML Listing 12 - Signed Advertisement XML schema

```
<xs:element name="SA" type="jxta:SA"/>
  <xs:complexType name="jxta:SA">
  <xs:sequence>
    <xs:element name="PSECred" type="jxta:PSECred"/>
    <xs:element name="jxta:Signature" type="base64binary"/>
    <xs:element name="jxta:Advertisement" type="base64binary"/>
  </xs:sequence>
</xs:complexType>
```

On regards to advertisements, JXTA does not currently seem concerned with

passive attacks, since it offers no advertisement protection against them. They are freely exchanged between peers in plain text. In fact, since they are structured using XML, it is very easy for an eavesdropper to read search for specific content (no need to process binary structured data). A human being can directly interpret advertisements with a text editor.

No effort is made either in order to masquerade advertisement traffic, so it is feasible that an attacker may obtain some interesting information by just analyzing traffic, specifically detecting which peers offer more resources (since they are the ones which publish more advertisements). Using this method, it is possible to find the most interesting peers when looking for potential victims to attack. In fact, since anybody may instantiate any peer group, acting as an interloper, and then discover advertisements bound to the group, this kind of attacks are easy to perform. It is not even necessary to tap the network. How easily advertisements are exchanged is both a bonus for open services and a bane for tight security environments.

If we assume that applications which use PSE correctly set a trust anchor in order to guarantee certificate authenticity, then active attacks such as spoofing, MitM and replay attacks may be correctly countered by digitally signing advertisements. Using this method, false advertisements may still occur within the peer group, but because of the non-repudiation property of digital signatures, it is easy to pinpoint offenders.

As far as resource publication is concerned, since every peer is completely reliant on its rendezvous peer in order to properly distribute the advertisement index to the rest of the network, and no control is made about which peer may become a rendezvous one, it is possible to pull off MitM attacks by masquerading as a one. No control mechanisms exist either to automatically detect a misbehaving rendezvous peer. For that reason, each application should always deploy some method in order for peers to be able to identify real rendezvous peers.

Finally, since secure advertisements lose the signature when stored into the local cache, the threat of a local attacker still exists, since it is possible to modify the local cache content, inserting or modifying false advertisements which redirect group members to false services in malicious peers.

6.3.4 Message exchange

In the current JXTA reference implementation, messaging has been secured under the assumption that the Personal Security Environment (PSE) has been chosen as a group's Membership Service. By using the PSE, JXTA messages may be secured at two different levels in core messaging: at the messaging level, by using the CBJX [Bai02] protocol, and at the wire transport level, via its own definition of TLS [Die99]. The messaging level operates at a higher abstraction level, encapsulating data without knowledge of the real network topology. The encapsulated data is then sent via the transport level, which does take into account both network topology and available transport protocols at the peer's node.

The standard messaging level provides the capability to include any type of digital signature elements into messages to be sent across the network. However, current standard messaging protocols never make use of this feature. CBJX (Crypto-Based JXTA Transfer) is a JXTA-specific protocol which provides lightweight secure message source verification by including into messages its own self-defined digital signature element, providing data integrity and authentication. This approach provides protection against active threats.

Even though CBJX is formally specified as a wire transport protocol, it can be truly considered to operate at the messaging level, or, more exactly, at a meta-messaging level. The main reason is that it lacks the capability to directly send messages between endpoints, which is what ultimately defines a wire transport protocol in JXTA. CBJX pre-processes messages in order to provide an additional secure encapsulation, creating a new message that is then relayed to an underlying wire transport protocol. For that reason, we classify CBJX as message level security.

In addition to the original message's digital signature, an information block, according to the definition shown in XML Listing 13, is also encapsulated with the secured message: a `CbJxMessageInfo` element, which contains the source peer credential (a PSE certificate), both the source and destination addresses, and the source peer ID.

This cryptographic information block is digitally signed as well, generating two distinct signatures within the final CBJX message. The certificate inside the cryptographic information block is used to validate both signatures.

In order to generate both signatures, XML data is serialized and then fed

XML Listing 13 - CBJX crypto-information XML schema

```
<xs:complexType name="cbjx:CbJxMessageInfo">
  <xs:sequence>
    <xs:element name="PeerCert" type="xs:base64binary"/>
    <xs:element name="DestinationAddress" type="xs:string"/>
    <xs:element name="SourceAddress" type="xs:string"/>
    <xs:element name="SourceID" type="jxta:JXTAID"/>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="JXTAID">
  <xs:restriction base="xs:anyURI">
    <xs:pattern value="([uU][rR][nN]:
                        [jJ][xX][tT][aA]:).+\-.+"/>
  </xs:restriction>
</xs:simpleType>
```

to the signature algorithm, processed as plain text. An overview of the final message encapsulation is shown in Figure 6.3. CBJX encapsulates signatures by using a single `Signature` element containing a Base64-encoded PKCS#7 [Kal98a] binary signature. Once the final CBJX message is complete, it is sent using any wire transport protocol, just like as a standard message.

On reception, the CBJX information block is extracted and both signatures are validated, acting in a transparent manner as far as upper layer protocols is concerned by providing the original message.

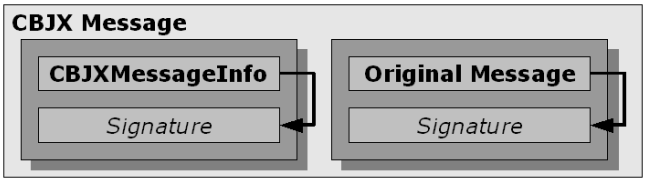


Figure 6.3: *CBJX secure encapsulation*

Apart from digital signatures, CBJX provides an additional lightweight authenticity method by using CBIDs. The concept of statistically unique and cryptographically verifiable IDs, has already been explained in Chapter 2.

At wire transport level, JXTA provides its own definition of standard TLS, allowing private, mutually authenticated, reliable streaming communications. Thus, TLS protects against both passive and active threats. As a wire transport proto-

col, it is responsible for encoding message data and sending it across the network.

The protocol is composed of two layers: the TLS Record Protocol and the TLS Handshake Protocol. The TLS Record Protocol provides connection security with two basic properties:

- The connection is private. Symmetric cryptography is used for data encryption (e.g., DES [FIP77], RC4 [Kau99], etc.) The keys for this symmetric encryption are generated uniquely for each connection and are based on a secret negotiated by the TLS Handshake Protocol. The Record Protocol can also be used without encryption.

- The connection is reliable. Message transport includes a message integrity check using a keyed MAC. Secure hash functions are used for MAC computations. The Record Protocol can operate without a MAC, but is generally only used in this mode while another protocol is using the Record Protocol as a transport for negotiating security parameters.

In the specific case of JXTA, messages are delivered securely between endpoints even when multiple hops across peers are necessary.

Even though TLS is a binary protocol, JXTA implements some of its data exchanges using XML elements (which encompass binary content). Three element types are defined in order to implement the protocol: TLS Content, which encapsulates transmitted secure data, Acknowledgements, which acknowledge data reception, and Retries, when a message is sent because of an apparent failure at a previous transmission. The latter element will be always present with a TLS Content element. All standard binary data structures defined in TLS are included into the TLS Content element.

By combining both CBJX and TLS, it is possible to trump both passive and active attackers by achieving data privacy, integrity and authenticity. Application developers may decide which protocol to use depending on their constraints (such as operating under a non-PSE peer group). It also must be taken into account that both types of transport methods do not support full advertisement propagation, they only support point-to-point communications. That means that applications which are based on multicast are still prone to security threats.

Service access

When accessing a service peer credentials must be checked in order to decide whether some peer has real access to that service. This is necessary since, as we could see in subsection 6.3.2, actually everybody may instantiate a peer group and try to access resources, acting as an interloper. This may be achieved in JXTA by using the peer group Access Service.

As far as the Access Service is concerned, the current JXTA reference implementation offers three kinds of access control, each one bound to each different membership service credential type:

The *Always* Access Service, which does not really check for access control and allows any operation. It is the default Access Service for peer groups.

The *simpleACL* Access Service uses Access Control Lists in order to establish which identities may perform each group operation. The access lists are distributed as parameters within the peer group advertisement.

The *PSE* Access Service provides an interface to PKIX certificate path validation. A trust anchor is set for the validation process and all credentials are validated against this anchor in order to decide whether the operation is permitted or not.

It must be pointed out that current Access Service approaches are strictly tied to ensure that some identity may access some service. Whether that identity really belongs to a legitimate peer group member is never checked, it is always assumed correct. Since the Membership Service is not up to the task of checking group membership either (any peer may claim any identity), as exposed in Section 6.3.2, this is something JXTA application developers should heavily take into account. However, it is possible to implement an Access Service which also checks group membership, as it is the case of the proposal presented in Chapter 5.

Furthermore, the Access Service provides a single primitive which just checks credential content, but does use on any kind of authentication protocol. This is not sufficient to guarantee protection against spoofing, since credentials are freely exchanged across the network, being public. Some other method must exist which tests credential authenticity, such as TLS or CBJX, in order to guarantee authenticity.

6.3.5 Disconnection

No real vulnerabilities threaten disconnection, apart from those which force an unintended shutdown due to unauthorized local access to the application. However, such problems are related to the operating system, so it can be considered outside the scope of this study. Disconnection was mainly included for the sake of completeness in formalizing the peer's full lifecycle.

6.3.6 Security evaluation summary

Table 6.2 summarizes the JXTA security evaluation, as far as core functionalities is concerned (the non-core proposals presented in Section 6.3.2 are not included). For each JXTA basic operation, it is shown whether it is vulnerable to each of the typical threats and which security mechanism exists in order to counter it. The threats are those listed in Section 6.2.2: Eavesdropping (Evs), Traffic Analysis (TAn), Spoofing (Spf), Main-in-the-Middle (MitM), Replay attacks (Rp), Local data alteration (LDA) and Software security flaws (SSF).

Operation\Threat	Evs	TAn	Spf	MitM	Rp	LDA	SFF
Startup	N/A	N/A	N/A	N/A	N/A	V(1)	P(OSS)
Join	N/A	N/A	V(5)	N/A	N/A	P(Key Enc.)	P(OSS)
Publish/Discover	V(2)	V(3)	P(Signed Adv.)			V(4)	P(OSS)
Messaging	P(TLS*)	V(3)	P(TLS*/CBJX)			V(4)	P(OSS)
Disconnect	N/A	N/A	N/A	N/A	N/A	N/A	P(OSS)

Table 6.2: *Security in JXTA summary (N/A: Non-applicable. V(type): Vulnerability exists. P:Protected(mechanism). *Not usable for message propagation)*

According to Table 6.2, the main vulnerabilities in JXTA's basic peer operations may be summarized in five different types as follows:

V(1): Code signing may be easily compromised. Malicious executable code can easily be built and cannot be automatically discovered when installed.

V(2): No encryption mechanism exists. Advertisements are transmitted in plain text.

V(3): No data flow masquerading mechanism exists. It is easy to identify important peers by its traffic.

V(4): No integrity check is enforced on the local cache. Changes by a local attacker are not discovered.

V(5): No real authentication is enforced. Any peer may ultimately join any group as an interloper.

6.4 Chapter summary

Being an OSS project, JXTA has been intensely reviewed, and as a result, its security features have improved over time. It can be summarized that the current implementation of JXTA has evolved to include an acceptable level of security, fulfilling minimum requirements for present day applications. However, this is at the cost of being bound to a very specific group membership model: PSE. In case that a custom model is chosen for some applications, most of its security capabilities may not be directly used, only CBJX becomes still available. This is not always desirable in a framework that was conceptualized to be open and easy to adapt to any environment. It would be useful that any custom application security model could make use of as many as possible JXTA secure mechanisms such as TLS or advertisement signature. Another useful feature, right now constrained by the assumption that PSE will always be used, would be the capability to use different types of keystores, apart from that in each peer's the local cache. Specially, being able to go beyond using the file system as cryptographic storage. However, it is not possible in the current JXTA specification.

It is also important to take into account when designing JXTA applications that, even though PSE provides a certificate based secure environment, it is still necessary to chose some methods in order to guarantee key authenticity. PSE assumes no trust model, just provides the necessary tools in order to deploy it. Furthermore, PSE makes no effort to provide any method of group membership revocation.

Finally, core JXTA operations still have some security gaps pending to be filled even when all its security capabilities are fully used (see Table 6.2). First of all, no mechanism exists in the current version of JXTA in order to secure messaging for propagation mechanisms, specially one that provides some degree of privacy.

In addition, no security exists for the local cache, even though very important data is stored inside. At least, some degree of integrity would be desirable, such as maintaining advertisement signatures.

Chapter 7

Securing JXTA core functionalities

The security review in Chapter 6 shows that the current JXTA reference implementation provides some degree of security to basic operations, such as using CBJX or TLS to secure messages and Signed Advertisements to secure advertisements. However, as shown in the summary Table 6.2, the provided methods are not fully satisfactory, since they are still open to some threats, mainly eavesdropping (Evs) and local data alteration (LDA). Furthermore, even under those cases where a security mechanisms exists, there are some non-trivial constraints, such as relying on the existence of a party that must be trusted by all peer group members via the PSE Membership Service, or being unable to provide privacy on multicast messaging. Finally, it is also a bit surprising that security mechanisms do not comply with the JXTA v2.0 specification's ideary of XML data formatting.

This chapter presents a method for providing two flavors of security to JXTA core functionalities related to publish/discovery and messaging, specifically core protocol messaging [AM09b] and advertisement distribution [AM08c]. This approach is suited to the idiosyncrasies of JXTA with the help of XML security standards: XMLdsig and XMLenc [W3C02a]. XMLdsig provides authenticity and non-repudiation, whereas XMLenc provides data privacy. Applications may choose to deploy any combination, since they have been designed in a modular way and nicely integrate.

This chapter is organized as follows. First of all, Section 7.1 provides an overview of the basic operations to be improved, expanding upon the explanation

presented previously in Chapter 6 and summarizing its main problems. Section 7.2 describes a proposal to protect core functionalities against local data alteration (LDA) via XMLdsig based messages, thus closely adhering to the JXTA ideary of XML formatting. Following, Section 7.3 describes how to achieve protection against eavesdroppers (Evs) using XMLenc, in such a way that the constraints in current mechanisms are avoided. The proposed mechanism also provides a method to control access to the published resources to a specific set of peers. Section 7.4 presents how to achieve key distribution, in order to properly deploy secure core functionalities. Finally, Section 7.5 summarizes the chapter's most important points.

7.1 Basic operations

Any application operating within a JXTA network exchanges information with other peers during the publish/discover and messaging basic operations, as described in Section 6.2.1. Since both operations are mainly concerned with data exchanges, they rely on JXTA's core messaging functionalities: the core protocols and the set of different advertisement types, respectively. In that sense, JXTA acts as an abstraction layer, so application developers need not to concern on how the JXTA core explicitly works. Being the gateway to all network communications, in the case that core messaging is somehow subverted, the network will stop functioning properly.

From the security analysis in Chapter 6, the current security mechanisms in JXTA's core functionalities are still vulnerable to some network threats, such as local data alteration attacks or eavesdropping. Furthermore, all security mechanisms are constrained to use the PSE Membership Service to manage peer groups. PSE provides an integrated secure environment in JXTA, but for some applications it may become too restrictive by restricting the peer group trust model to one based on a TTP and forcing the use of X.509 certificates. The use of a TTP is not always desirable in a dynamic and decentralized environment such as P2P, specially when trying to maximize peer equality and self-organization. Some additional problems which increase the system's complexity are also inherited, such as certificate chain management and revocation.

7.1.1 Publish and Discovery

Resource publication and discovery via JXTA advertisements was thoroughly reviewed in Section 6.1.3. Advertisements are metadata documents formatted in XML, exchanged between peers using the JXTA core protocols, describing a resource and how it may be accessed. Peers cannot access a resource without previously retrieving its associated advertisement. They are specially sensitive to attacks for several reasons. First of all, it is very easy to propagate false information, since rendezvous peers efficiently distribute such information across network boundaries. In fact, since rendezvous peers only store indexes, they might not even be aware that false information is being distributed. Furthermore, in the case that two peers cannot directly communicate due to connectivity issues, such as the use of NAT or a firewall, advertisements are routed through rendezvous peers. Therefore, rendezvous peers have access to advertisements before they reach the final destination, becoming a potential point of failure. Finally, JXTA allows any peer to store any advertisement, even in the case that such peer is not its original creator.

One of the most obvious security threats in this environment is the possibility of spoofing peer identifiers. Peers may publish bogus resources with random identifiers (or even worse, some other peer's identifier), creating a denial-of-service attack on the network by filling it with rubbish. It is not possible to avoid the publication itself, since the communication channel is always open, but it should be possible to quickly recognize which peers are trying to actively disrupt the network, in order to isolate them or expel them from the peer group.

Another security threat in this environment is the possibility that an unauthorized peer obtains some resource's associated advertisement, and thus is able to access it henceforth. This is a big concern since in JXTA advertisement may be stored or transmitted across peers which are not its original creator, for example, in order to provide redundancy. In this scenario, even though it is not possible to control which peers have access to the advertisement itself, it is feasible to control which ones will be able to effectively use its content. In that case, a trade off between data privacy and accessibility should be reached, since it must still be possible for core discovery services to locate and distribute advertisements within the network.

In addition to the shortcomings generic to core functionalities, JXTA adver-

tisements are specially susceptible to transient security. Advertisements are only secured during transport. Since they are stored into the peers' local cache until their expiration, they should also be kept secure there. The transient nature of the security layer also forces applications to verify all messages upon immediate reception, before secure encapsulation is discarded, even though most of them might never be used, thus impacting performance. Furthermore, it is not possible to effectively store advertisements in other peers, but deny them access to some of its fields. This is not a fringe scenario, since it happens whenever advertisements are pushed to third parties in order to guarantee availability or redundancy, such as in remote publication or when the receiving peer is acting as a rendezvous.

7.1.2 Messaging

JXTA defines a set of six core protocols specifically suited for ad hoc, pervasive, multi-hop, P2P computing. JXTA's core protocols allow peers to cooperate and form autonomous peer groups transparent to their location, as well as providing the necessary services in order for any other protocol to be used in JXTA applications to operate within the network. Peers may use such protocols in order to advertise and discover resources, join peer groups and dynamically route messages across multiple network hops. They also make few assumptions about the underlying network transport, in order to guarantee that they may be applied to the broadest set of network scenarios.

JXTA's six core protocols are: Peer Discovery Procool (PDP), Peer Information Protocol (PIP), Peer Resolver Protocol (PRP), Pipe Binding Protocol (PBP), Endpoint Routing Protocol (ERP) and Rendezvous Protocol (RVP). Their purpose was already described in Section 6.1.2.

It is not mandatory for JXTA implementations to deploy all core services, but at least PRP and ERP must be supported in order to provide addresses to peers and allow communication between endpoints. The remaining protocols are optional, but supporting them increases interoperability and provides a wider degree of functionality. The current JXTA J2SE implementation [jxt07] supports all six of them.

As it is shown in Figure 7.1, JXTA endpoint communication is structured in a classical layered approach, the core protocols acting as a gateway to networking operations under a P2P environment. This provides an abstraction layer to both

JXTA's own services and custom made application dependant ones (operating at the upper Services layer), enabling the deployment of services in a transparent manner to the real underlying transport methods or topology.

At the Peer layer, the higher level core protocols (PIP, RVP, PBP and PDP) allow services to publish, locate and exchange resources. The Endpoint layer manages routing and addressing, via the ERP protocol, and specifies the format for all query-response exchanges, using the PRP protocol. This means that all core protocols' queries sent across the network are ultimately encapsulated into a PRP query. PRP queries are then encapsulated as messages at the Messaging layer. Finally, the message is sent across the network using any of the wire transport protocols (chosen according to the application's needs) at the Wire Transport layer, such as TCP, HTTP, multicast or TLS. The message generated at the Messaging layer is considered the application level data to be sent by the wire transport protocol.

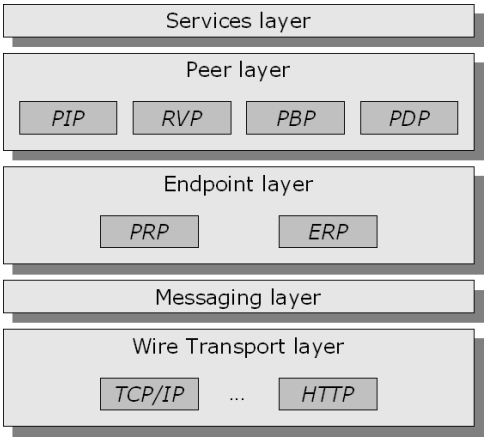


Figure 7.1: *JXTA protocol layers*

A message is a set of named and typed content elements, which means that it is essentially a set of name/value pairs, organized as an ordered sequence. The content element can be an arbitrary type. The most recently added element appears at the end of the message. As a message passes down each layer, one or more named elements may be added to the message. As a message passes back up the stack, each layer will remove these elements.

All core protocols are codified using XML, each message element name being the root XML element tag and its content the corresponding XML subtree. The main reasons for such format are its programming language/platform independence, being self-describing and being able to enforce correct syntax. As an additional feature, XML can easily be translated into other encodings, such as HTML, which may allow peers that do not support XML to access resources. The XML schema for a PRP query is shown in Listing 14 as a sample of core protocol format. It is an interesting election, since all Peer layer protocols use PRP to transmit messages (by encapsulating its own content into the `Query` element).

XML Listing 14 - PRP Query schema

```
<xs:complexType name="ResolverQuery">
  <xs:sequence>
    <xs:element ref="jxta:Cred" monOccurs="0"/>
    <xs:element name="SrcPeerID" type="jxta:JXTAID"/>
    <xs:element name="HandlerName" type="xs:string"/>
    <xs:element name="QueryID" type="xs:string"/>
    <xs:element name="HC" type="xs:unsignedInt"/>
    <xs:element name="Query" type="xs:anyType"/>
  </xs:sequence>
</xs:complexType>
```

Apart from the issues described at the beginning of this section, core protocols have the additional shortcoming of no privacy at the messaging layer. Currently, the only way to achieve data privacy, in order to counteract passive attacks such as eavesdropping, is using TLS. However, since TLS is a wire transport protocol, it imposes a constraint that cannot be ignored: no other transport protocol may be used underneath. That means that it is not possible to support JXTA's message propagation via multicast, as well as plain HTTP proxies either. Furthermore, as new transport protocols become supported in JXTA (for example, UDP or RTP), it will not be possible to use them in a secure manner. Finally, endpoints must execute an agreement protocol previous to message exchange, vanishing any advantage provided by the asynchronous approach of core protocols. In a P2P environment, a fire-and-forget approach to messaging is much more desirable.

7.2 Protection against local data alteration

In this section, we present security mechanisms to achieve protection against local data alteration, a vulnerability issue identified in Table 6.2. Our proposal takes into account all the previously exposed issues to secure data exchange at both core protocol and advertisement level. An extension format based on XMLdsig is defined for both PRP messages and advertisements, but maintaining their base structure according to the JXTA v2.0 protocol specification. This approach even allows peers which do not support signatures to process the message content nevertheless. Peers may freely choose which data should be secured, and chose its own degree of security, without impacting other peers. On the other hand, the use of a TTP is avoided by using a method based in CBIDs described in Chapter 5.

Using XMLdsig is a logical approach, since it is a standard for signing XML data and all protocols in JXTA are XML-formatted. Apart from keeping message readability, XMLdsig offers some additional capabilities which are important in this environment.

On one hand, it maintains interoperability by taking into account XML canonicalization. This is extremely important when using XML, since documents which are syntactically different may translate as semantically equal (for example, changing the order of sibling XML elements). Directly feeding XML data to a signature algorithm does not take this fact into consideration. Therefore, just using a different XML parser may change the processed data, however irrelevant to the XML semantics, and will invalidate a signature. Just for that only reason, XMLdsig is better when signing XML data.

On the other hand, XMLdsig is an open specification which allows the definition and inclusion of new types of credentials in order to transport the public keys which validate the signature. Therefore, it is easy to integrate with any credential-based membership service, including PSE or the one specified in Chapter 5.

7.2.1 Core protocol integrity and authenticity

Core protocol messaging integrity and authenticity is achieved through a detached signature within the message body, as shown in Figure 7.2. The XML signature is included as a message signature, just as exposed in Section 6.3.4, but instead of

a self-defined single `Signature` element, a full XMLdsig signature is included. In contrast with CBJX, no additional encapsulation is needed, since the XML signature contains all needed information related to the security layer. As a result, messages generated using this method may be processed even by peers which do not support signatures (they are able to decode the original message and ignore the signature).

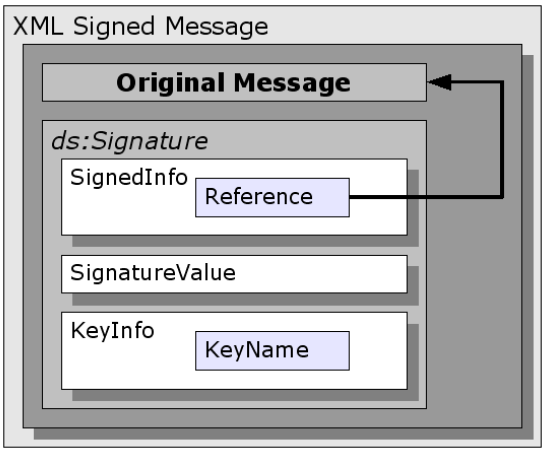


Figure 7.2: *XMLdsig detached signature profile*

As a detached signature, in this scenario it is enough to use a default URI in the `Reference` element in order for the Messaging layer to locate the corresponding signed data (the original message). The `KeyName` element will be used to retrieve the signer's public key, in order to validate the signature, as will be shortly explained in Section 7.4.

A sample message signature is shown in Listing 15 (some ID's and Base64 encoded data have been shortened). Notice that, since it is a detached signature, it is not necessary for the message to be present.

7.2.2 Advertisement integrity and authenticity

The signed advertisement generation profile, on the other hand, uses an enveloped signature approach to provide integrity and authenticity. The XML `Signature` element is appended to the original advertisement as an additional element under

XML Listing 15 - Signed message (detached signature)

```
<ds:Signature
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm=
        "http://www.w3.org/TR/2001/REC-xml-c14n-20010315">
    </ds:CanonicalizationMethod>
    <ds:SignatureMethod Algorithm=
        "http://www.w3.org/2000/09/xmldsig#rsa-sha1">
    </ds:SignatureMethod>
    <ds:Reference>
      <ds:Transforms>
        <ds:Transform Algorithm=
            "http://www.w3.org/2001/10/xml-exc-c14n#">
        </ds:Transform>
      </ds:Transforms>
      <ds:DigestMethod Algorithm=
      "http://www.w3.org/2000/09/xmldsig#sha1">
      </ds:DigestMethod>
      <ds:DigestValue>Ko0R31wMpcJ17VAmtaUf7nS/KU4=
      </ds:DigestValue>
    </ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue> nloCRUg4WB0H+DcEAuLKGYhqvsfdRCy4R...
    ...QYH8Czizo3P AkvLI1UGoMekOHRL2kI=
  </ds:SignatureValue>
  <ds:KeyInfo>
    <KeyName>urn:jxta:uuid-596162...BCF0646C103</KeyName>
  </ds:KeyInfo>
</ds:Signature>
```

the root document node. A sample signed Peer Group Advertisement is shown in Listing 16 (some ID's and Base64 encoded data have been shortened).

The main benefits of maintaining the advertisements base format, in contrast with the original JXTA Signed Advertisement, is the fact that it is no longer necessary to pre-process data at the reception time before being stored into the cache. The full signed advertisement content is readily accessible to indexing services at any time. Furthermore, peers which do not support signed advertisements are also able to process them, allowing the deployment of heterogeneous networks, where peers may decide to support different security degrees, but can still understand each other.

Whenever some peer receives a signed advertisement, it is stored into that peer's cache, its verification deferred up to the moment the advertisement is actually used. This feature may improve performance, since there may be a lot of advertisement traffic in a given network that the peer will not use and that it does not need to be validated. This method maintains end-to-end advertisement security for its whole lifetime, not just during transport, keeping its security layer

XML Listing 16 - Signed Peer Group Advertisement

```
<jxta:PGA xmlns:jxta="http://jxta.org" xml:space="preserve" ID="signed">
  <GID>urn:jxta:uuid-2AD...5F02</GID>
  <MSID>urn:jxta:uuid-EB76C91C2A0F49F685C...B3812F206</MSID>
  <Name>SampleGroup</Name>
  <Desc>Signed Peer Group Advertisement</Desc>
  <ds:Signature
      xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm=
          "http://www.w3.org/TR/2001/REC-xml-c14n-20010315">
      </ds:CanonicalizationMethod>
      <ds:SignatureMethod Algorithm=
          "http://www.w3.org/2000/09/xmldsig#rsa-sha1">
      </ds:SignatureMethod>
      <ds:Reference URI="signed">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/
              2000/09/xmldsig#enveloped-signature">
          </ds:Transform>
          <ds:Transform Algorithm=
              "http://www.w3.org/2001/10/xml-exc-c14n#">
          </ds:Transform>
        </ds:Transforms>
        <ds:DigestMethod Algorithm=
        "http://www.w3.org/2000/09/xmldsig#sha1">
        </ds:DigestMethod>
        <ds:DigestValue>Ko0R31wMpcJ17VAmtaUf7nS/KU4=
        </ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue> nloCRUg4WB0H+DcEAuLKGYhqvsfdRCy4R...
      ...QYH8Czizo3P AkvLI1UGoMekOHRL2kI=
    </ds:SignatureValue>
    <ds:KeyInfo>
      <KeyName>urn:jxta:uuid-596162...BCF0646C103</KeyName>
    </ds:KeyInfo>
  </ds:Signature>
</jxta:PGA>
```

even when stored in a peer's local cache, until expiration. Even in the case that the peer's cache is compromised by a local attacker, it is possible to detect data alteration

7.2.3 Signed data processing

The signature process is straightforward, since the signer holds all the necessary information in order to create a signed advertisement: its private key and the original data.

At the moment of validation, the following steps are performed:

1. Retrieve the source peer public key (see details in Section 7.4).

2. Apply the SHA-1 hash algorithm to the public key. Generate a JXTA CBID from the result, as described in Chapter 5.

3. Compare the resulting CBID to the source peer CBID. If equal, key authenticity is proved. Otherwise, the process stops.

4. Validate XML signature using the public key retrieved in Step 1. If valid, integrity, authenticity and non-repudiation are proved.

In addition, by using CBIDs as the mechanism for key authenticity, no TTP is necessary, providing lightweight key authenticity.

7.3 Protection against eavesdropping

JXTA provides no real data privacy at advertisement level, only at transport protocol level, resulting in the heavy constraint that no privacy may be provided to propagation. Furthermore, it relies on an all-or-nothing approach: either peers have access to the full data or they cannot access it. There is no possibility to provide partial access to content. This is a very useful feature in the particular case of advertisements, where access to service fields may be limited to some subset of peers.

In order to solve all the previously exposed constraints, we define an additional extension format for advertisements based on XML encryption, also maintaining the base structure of advertisements as defined in the JXTA v2.0 protocols specification. It is possible to choose which fields are secured and which peers may access each field, instead of being an all-or-nothing approach. By choosing which fields to secure, it is possible to control which peers may access specific services announced via the advertisement.

At advertisement level, secured advertisements may be freely distributed and still remain encrypted when stored in other peers' local cache. Peers which do not support encryption may nevertheless process the clear text fields in a transparent way. Peers may freely choose which advertisements should be secured, and chose its own degree of security, without impacting other peers. CBIDs are still used to avoid the need for a TTP.

In this case, exactly the same XMLenc profile is used for both core protocols messages and advertisements.

7.3.1 Selective data encryption

In contrast to XML signature, there are several approaches to XML document encryption [Had02; GP02; W3C02a]. For our proposal, we specifically use XMLenc, the main reasons being its status as an XML standard, its flexibility and its capability to guarantee data privacy during transit or when stored in parties different from the one which generated the document, which is not supported in [Had02]. XMLenc also provides a reasonable result document size, in contrast with [GP02], and it is the easiest to integrate with approaches which provide additional security services by using XMLdsig.

XML fields are selectively encrypted using a wrapped key encryption scheme (such as the one defined in [Kal98b]). For each advertisement field to be encrypted, a symmetric key is generated and used to encrypt the field. The symmetric key is then encrypted (wrapped) using each recipient's public key, obtaining a set of encrypted keys, each one of which can only be accessed by one group member.

This scheme is applied to peer advertisements or PRP messages according to the schematics shown in Figure 7.3. Each sender may choose which fields will be encrypted. The encryption profile takes into account that all of JXTA's XML documents may contain three distinct types of fields, depending on the type of metadata stored within. Some constraints must be followed when selecting which fields are encrypted.

Mandatory fields must always exist in an advertisement, usually resource identifiers (such as the GID and MSID fields). This fields are necessary to properly propagate PRP messages or locate the advertisements via the Discovery Service. For that reason they should never be encrypted.

Optional fields provide additional information, but are not mandatory. They can be freely encrypted (or only some subset of them), but then services which rely on them will not be able to process them.

Service fields define specific services or resources, serving as the means to accessing them. In the case of PRP core protocol messages, only one of such field exists, which contains the whole transmitted data. If this field is encrypted, only the selected subset of peers which may decrypt the data will be able to properly access the resource. They usually take the form of a *Svc* XML element in Peer and Peer Group Advertisements, and a *Query* or *Response* XML element for PRP queries and responses, respectively.

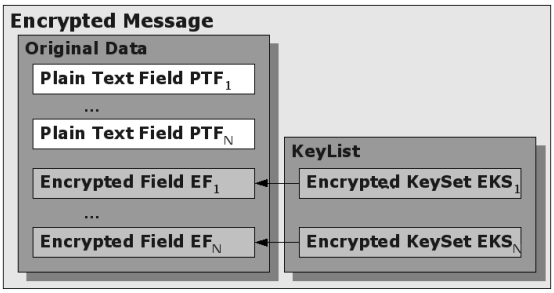


Figure 7.3: *XML data encryption schematics*

Wrapped keys are included by introducing an additional field: `KeyList`. This field follows the same syntax as any other XML field. Each encrypted field is associated to a set of a wrapped keys included in the `KeyList` field. Such set contains an entry for each destination peer which should be able to access the encrypted field. Encrypted fields with exactly the same destination peers will be associated to the same key set, effectively sharing it.

A wrapped key is defined by an XMLenc `EncryptedKey` element and contains all the cryptographic information necessary to decrypt such field. In this manner, XMLenc data can be appended to advertisements without invalidating its schema definition, meaning that they may be integrated into JXTA in a transparent way, maintaining interoperability even when encrypted advertisements are used.

Encrypted fields are defined by XMLenc `EncryptedData` elements. Figure 7.4 shows the encryption profile we define in order to link each `EncryptedData` element to its corresponding `EncryptedKey`. Peers that receive the XML document may identify which `EncryptedKey` fields may be decrypted with its own private key by searching for its Peer ID in the `KeyInfo` field of each `EncryptedKey` element. Every `EncryptedData` element may have several linked `EncryptedKey` elements, one for each peer which may access the service.

By using this XML profile, it is possible to accommodate selective entry encryption as well as supporting propagation. In addition, by keeping mandatory fields as plain text, a trade off is achieved between data privacy and the capability to use the Discovery Service in order to locate resources for the case of encrypted advertisements. Encrypted fields become invisible to its basic operation. The

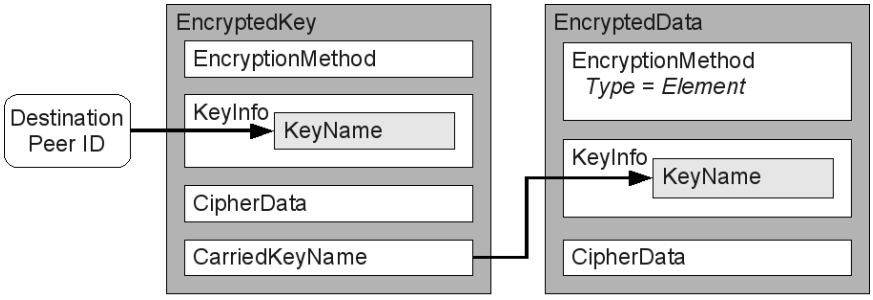


Figure 7.4: *XMLenc encryption profile*

detailed encryption/decryption process is described in subsection 7.3.2.

With this approach, apart from standard data privacy, it is now possible to provide access control to group services by encrypting specific advertisement fields: those which hold service definition and configuration parameters. Only those peers which can decrypt such fields will be able to ultimately locate the services. By using selective encryption, it is possible to choose which subset of peers may access each service.

7.3.2 Encryption and decryption process

The encryption process that generates the XMLenc profile can be described as follows.

Encryption:

1. Peer A decides to transmit some message (for example, publish an advertisement).

2. For each field (optional or service) defined within the message, peer A chooses the subset of peers P_i, for $i = 1, \cdots, n$, which will be able to access it. Fields with the same subset of destination peers are then grouped into FG_j, for $j = 1, \cdots, m$, field groups.

3. A new field, the `KeyList` field, is appended to the message.

4. For each field group FG_j, for $j = 1, \cdots, m$ is processed in the following manner:

   (a) Both a random symmetric key K_j and an identifier id_j are generated by A.

   (b) Each field $F \in FG_j$ is encrypted according to XMLenc with K_j. Each original field becomes an XMLenc `EncryptedData` element. id_j is appended to the each `EncryptedData` element, as a KeyInfo element.

   (c) For each peer P_i, for $i = 1, \cdots, n$:

      i. A retrieves PK_i, using the method described in Section 7.4.

      ii. Before encryption, A checks the authenticity of PK_i via CBIDs with the method described in Chapter 5.

      iii. K_j is wrapped (encrypted) using PK_i, generating an XMLenc `Encryptedkey` element. The `CarriedKeyName` field of this element is set to id_j. Its `KeyInfo` field is set to B's Peer ID by using a `KeyName` element as previously shown in Figure 7.4.

      iv. The `EncryptedKey` element is added to the `KeyList` field.

   (d) Once all peers in P_i, for $i = 1, \cdots, n$, have been processed, the `KeyList` field includes the wrapped keys for all peers P_i, for $i = 1, \cdots, n$.

5. An encrypted message has been generated according to the format defined in 7.3.1. For each encrypted field F, a set of wrapped keys exist within the `KeyList` field which may decrypt it. Some fields may share the same set of wrapped keys.

6. If the encrypted message is an advertisement, it is published via the core JXTA Discovery Service. Otherwise, it is transmitted using a chosen transport layer protocol.

A sample encrypted Peer Group Advertisement after this process is shown in Listing 17 (some ID's and Base64 encoded data have been shortened in order to improve readability). It contains two original service fields (the `Svc` elements), but only one of them, S_1, has been encrypted. The field entry in the encrypted advertisement corresponds to the `KeyList` field, which contains the wrapped keys necessary in order to decrypt S_1. In this example, only two peers (with Peer

ID *urn:jxta:uuid-59...B6A403* and *urn:jxta:uuid-2B...D0A603*) may properly decrypt S_1. The identifier used to associate the encrypted field to the set of wrapped keys (encryption process, step 5a) is *KeyId-0*.

XML Listing 17 - Selectively encrypted Peer Group Advertisement

```
<?xml version="1.0" encoding="UTF-8"?>
<jxta:PGA xmlns:jxta="http://jxta.org" xml:space="preserve">
  <GID>urn:jxta:uuid-E7...02</GID>
  <MSID>urn:jxta:uuid-F2...06</MSID>
  <Name>My Peer Group Name</Name>
  <Desc>My Peer Group Description</Desc>
  <Svc>
    <xenc:EncryptedData xmlns:xenc="...xmlenc#" Type="...xmlenc#Content">
      <xenc:EncryptionMethod Algorithm="...xmlenc#aes128-cbc"/>
      <ds:KeyInfo xmlns:ds="...xmldsig#">
        <ds:KeyName>KeyId-0</ds:KeyName>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>iTXqXYoONtoxB44J...VUDg==</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </Svc>
  <Svc>
    <MCID>urn:jxta:uuid-5326C14064F24A5DB634D14A1118ED0F05</MCID>
    <Parm type="myapp:MyServiceConfig">
      <MyParameter1>My Value 1</MyParameter1>
      <MyParameter2>My Value 2</MyParameter2>
    </Parm>
  </Svc>
  <KeyList>
    <xenc:EncryptedKey xmlns:xenc="...xmlenc#">
      <xenc:EncryptionMethod Algorithm="...xmlenc#rsa-1_5"/>
      <ds:KeyInfo xmlns:ds="...xmldsig#">
        <ds:KeyName>urn:jxta:uuid-59...B6A403</ds:KeyName>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>ip1y...QxI=</xenc:CipherValue>
      </xenc:CipherData>
      <xenc:CarriedKeyName>KeyId-0</xenc:CarriedKeyName>
    </xenc:EncryptedKey>
    <xenc:EncryptedKey xmlns:xenc="...xmlenc#">
      <xenc:EncryptionMethod Algorithm="...xmlenc#rsa-1_5"/>
      <ds:KeyInfo xmlns:ds="...xmldsig#">
        <ds:KeyName>urn:jxta:uuid-2B...D0A603</ds:KeyName>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>m9Jm...vLM=</xenc:CipherValue>
      </xenc:CipherData>
      <xenc:CarriedKeyName>KeyId-0</xenc:CarriedKeyName>
    </xenc:EncryptedKey>
  </KeyList>
</jxta:PGA>
```

Decryption:

Whenever a peer $B = P_l$ for some $l = 1, \cdots, m$ wants to process some en-

crypted data:

1. Peer B locates an advertisement via the core JXTA Discovery Service or receives a message via a transport layer protocol.
2. B locates the `KeyList` field in the XML document.
3. B locates within the `KeyList` field the set of `EncryptedKey` elements, EK, which contain B's Peer ID in its `KeyInfo` field.
4. For each `EncryptedKey`, $EncKey_i$, in EK:
   (a) B selects the set of `EncryptedData` elements, ED, within the document which `KeyInfo` value match the `CarriedKeyName` value in $EncKey_i$. This value corresponds to the identifier id_j generated in step 5a of the encryption process.
   (b) B's private key is used to decrypt the symmetric key, K_i, stored in Enc_i.
   (c) Each `EncryptedData` element $EncData_i \in ED$ is decrypted using K_i.
   (d) The original field F has been recovered.
5. B obtains the final data where some service entries may still be encrypted. B has no access to such entries, only to those he could satisfactorily decrypt.

The use of XML encryption does not hinder signed advertisements or messages, since both XMLenc and XMLdsig validation nicely integrate. XML signature processing automatically detects that some signed data is encrypted, so it must be previously decrypted before signature validation. This is achieved by including an XMLenc `Transform` element within the signature `Reference` element.

It's worth mention that whenever a peer receives an encrypted advertisement, it can be stored into the peer's cache and its decryption can be deferred up to the moment the advertisement is actually used, just as it was the case for signed advertisements. Furthermore, just like signed advertisements, it maintains end-to-end advertisement security for its whole lifetime, not just during transport,keeping it secure even when stored in a peer's locally cache. Even though JXTA may use its current implementation of TLS in order to provide a message security layer,

such security is transient. Using this approach, it is possible to provide secure advertisement propagation, which is not currently possible.

In addition, the proposed method also takes special care to keep interoperability by maintaining as much as possible of the base message format in the encrypted format, public key wrapping and key distribution, instead of creating completely new metadata documents. Encrypted message interoperability is achieved by selectively encrypting fields. Mandatory fields can still be accessed and processed by services which rely on them, such as the Discovery Service (both locally or at the SRDI super-network).

7.4 Key Distribution

As a precondition for the new XML based messaging to function, it is necessary to properly distribute each peer's public key to the rest of group members. In order to properly distribute the key, it is possible to take advantage of JXTA's resource publication methods by using the Peer Advertisement, adding some additional fields. In this way, no extra message types or additional protocols are necessary, seamlessly integrating with the current capabilities of JXTA. Specifically, the service fields are used for public key transport and distribution. Those peers which do not support advertisement security will ignore the service fields, but they will still recognize Peer Advertisements as such.

Key authenticity is ensured by signing advertisements and using CBIDs as JXTA peer IDs, which also guarantees that active attacks do not subvert the original data. Signatures are generated using an encapsulated XMLdsig signature type.

A sample of the new format for a Peer Advertisement that enables public key distribution to a specific Peer ID is shown in Listing 18. The standard service fields (`Svc`) are used in order to encapsulate all public key related information. In this case, a raw RSA public key is being distributed. The advertisement is also signed in order to ensure key authenticity and data integrity.

Peer Advertisements will always contain an XML signature with a `KeyInfo` element which explicitly encapsulates the public key. Several key types are supported by using different subelement types: `KeyValue` (raw public key, used in the example), `X509Data` (X.509 certificate), `PGPData` (PGP key) and `SPKIData` (SPKI [Ell99] certificates). However, as mentioned before, the standard supports

the definition of new key transport types.

In order to retrieve the Peer Advertisement (and the corresponding public key), the source Peer ID can be obtained from contained `KeyName` element in signed messages. In the case of encrypted messages, it is obvious that the sender must already know the destination ID. With that ID, it is the possible to find the corresponding Peer Advertisement in the local cache. In the case that a peer did not previously receive the sender's Peer Advertisement and it cannot be found in the local cache, it may be easily retrieved by asking the sender or a rendezvous peer via the JXTA Discovery Protocol (specifically, by using a type 0 query). Using this protocol, any peer may retrieve any advertisement within the group.

Keeping the peer's public key in the Peer Advertisement is not inconvenient for signature or encryption processing since, if an advertisement exchange is happening, it is obvious that both peers are online. Nevertheless, JXTA provides a mechanism (remote publication) in order to push Peer Advertisements to any group member, providing redundancy.

Finally, it is important to remark that this advertisement just makes public such binding, but the binding itself is ultimately achieved via the use of CBIDs. Whenever a Peer Advertisement is received, the binding validity may be tested in order to ensure its correctness.

7.5 Chapter summary

This chapter has presented a new proposal for securing JXTA's core functionalities, providing privacy, integrity and authenticity using both XMLdsig and XMLenc. This proposal may be used in core protocols and advertisement distribution. The proposal maintains end-to-end advertisement security for its whole lifetime, not just during transport, keeping its security layer even when stored in a peer's localy cache, until expiration. Even though JXTA may use its current implementation of TLS or CBJX to provide security at transport layer, such security is transient and puts some constraints, such as the use of PSE or the inability to use propagation. With the proposal presented in this chapter, both constraints are solved. Furthermore, our proposal keeps interoperability by maintaining the data base format, instead of creating a completely new one. Therefore, it is possible to seamlessly integrate peers which support security with those who do not (or chose not to). With this proposal, some of the most important vulnerabilities

in current JXTA security, as described in Chapter 6 are solved.

Additionally, data privacy allows selective field encryption with multiple destinations, moving away from the current all-or-nothing approach. Peers have the possibility to define which sets of group members may access specific data. Using this method, it is possible to provide discretionary access control to published resources.

XML Listing 18 - Key distribution via Peer Advertisement

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jxta:PA>
<jxta:PA xml:space="preserve" xmlns:jxta="http://jxta.org">
  <PID>urn:jxta:uuid-59...B03</PID>
  <GID>urn:jxta:jxta-NetGroup</GID>
  <Name>My Peer Advertisement Name</Name>
  <Desc>My Peer Advertisement Description</Desc>
  <Svc>
    <MCID>urn:jxta:uuid-DE...05</MCID>
    <Parm>
    <jxta:RA><Dst><jxta:APA>
      <EA>tcp://...:9701</EA>
      <EA>cbjx://uuid-59...03</EA>
      <EA>jxtatls://uuid-59...03</EA>
      <EA>relay://uuid-59...03</EA>
    </jxta:APA></Dst></jxta:RA>
    </Parm>
  </Svc>
  <Svc>
    <MCID>urn:jxta:uuid-8C...05</MCID>
    <Parm>
  <ds:Signature
      xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm=
          "http://www.w3.org/TR/2001/REC-xml-c14n-20010315">
      </ds:CanonicalizationMethod>
      <ds:SignatureMethod Algorithm=
          "http://www.w3.org/2000/09/xmldsig#rsa-sha1">
      </ds:SignatureMethod>
      <ds:Reference URI="">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/
              2000/09/xmldsig#enveloped-signature">
          </ds:Transform>
          <ds:Transform Algorithm=
              "http://www.w3.org/2001/10/xml-exc-c14n#">
          </ds:Transform>
        </ds:Transforms>
        <ds:DigestMethod Algorithm=
        "http://www.w3.org/2000/09/xmldsig#sha1">
        </ds:DigestMethod>
        <ds:DigestValue>Ko0R31wMpcJ17VAmtaUf7nS/KU4=
        </ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue> nloCRUg4WB0H+DcEAuLKGYhqvsfdRCy4R...
      ...QYH8Czizo3P AkvLI1UGoMekOHRL2kI=
    </ds:SignatureValue>
    <ds:KeyInfo>
      <ds:KeyValue><ds:RSAKeyValue>
        <ds:Modulus>mHkQ53C6WU=</ds:Modulus>
        <ds:Exponent>AQAB</ds:Exponent>
      </ds:RSAKeyValue></ds:KeyValue>
    </ds:KeyInfo>
  </ds:Signature>
    </Parm>
  </Svc>
</jxta:PA>
```

Chapter 8

Conclusions

The conclusions of this thesis are briefly exposed in this chapter, summarizing all the presented research work.

8.1 Concluding remarks

This work has proposed several methods to secure a group based P2P environment from a peer lifecycle standpoint. This lifecycle can be divided into two different stages: peer group joining and operation. The former is concerned with proof of group membership, so a peer may be recognized by other group members. The latter deals with secure data exchange once a peer is considered a legitimate group member. Each stage has its own particular challenges that must be solved.

One of the the main differences between both stages is the fact that group membership can be modelled and analyzed using a more generic approach, whereas secure group operation depends on how each specific P2P middleware actually works. Therefore, a particular group based middleware was chosen: JXTA, a quite popular open protocol specification endorsed by SUN Microsystems. The maturity of its reference implementation is one of its best selling points.

Even though the quest for security is not the main goal for many systems, much more concerned with protocol efficiency and scalability, JXTA is quite sensitive to security matters, as can be concluded from this research work. However, the current specification of JXTA is still open to improvements. On regards to group

joining, it still relies on a TTP, a centralized approach which does not take into account peer equality. On the other hand, secure protocols are not consistent with its ideary of XML formatting and interoperability. Therefore, this thesis provides some insights about how to steer JXTA towards new directions which keep the P2P pure, taking into account an heterogeneous network.

8.2 Contributions

This thesis has contributed in providing a secure environment to peer group based P2P systems at two different levels: peer group membership and member interaction. For that reason, the thesis has been presented in two distinct, but clearly interrelated, parts. The main goal has been keeping the P2P model pure, distancing from approaches which, tough quite popular, ultimately rely on a common TTP, which results in backwards step on regards to network decentralization. In order to maintain a pure model, a web-of-trust based approach has been taken, capitalizing peer autonomy and self-organization. The P2P middleware JXTA has been chosen in order to specify and refine the proposed mechanisms.

In the first part, concerned with group membership, the group access control process has been formalized as a set of scenarios dependant on specific parameters, according to each group member's involvement degree. From these scenarios, it has been possible to specify which are the requirements that each approach to peer group membership must fulfill in order to solve them. As a result, the requirements of using web-of-trust for peer group membership have been identified, which led to the following publications:

- J. Arnedo-Moreno and J. Herrera-Joancomartí. "Identifying different scenarios for group access control in distributed environments". *Actas de la IX Reunión Española sobre Criptología y Seguridad de la Información (IX - RECSI)*. pp. 388–399. 2006

- J. Arnedo-Moreno and J. Herrera-Joancomartí. "Resolución de escenarios en control de acceso a grupo en entornos distribuidos". *Actas del II Simposio sobre Seguridad Informàtica, Congreso Español de Informática (II - SSI, CEDI)*. pp. 119–126. 2007

Next, a generic method for access control in peer groups using a web-of-trust is presented. Apart from keeping the P2P model pure and providing an alternative

to TTP based approaches, the other main contributions are twofold. First of all, its ability to adapt to a broad range of group policies and group membership scenarios, providing necessary capabilities in order to be more restrictive or open when needed. In addition, the proposal minimizes de length of certification paths in order to avoid the validation of long certification chains, which is the main problem in current methods. Additionally, the provided method may integrate extra desirable features such as peer anonymity. This work led to the publication of:

- J. Arnedo-Moreno and J. Herrera-Joancomartí. "Providing a collaborative mechanism for peer group access control." In *Proceedings of the Workshop on Trusted Collaboration (TrustCol'06)*, pp. 1–6. IEEEPress, 2006.

- J. Arnedo-Moreno and J. Herrera-Joancomartí. "Maintaining unlinkability in group based P2P environments." In *Proceedings of the 5th International Workshop on Collaborative Peer-to-Peer Systems (COPS'09)*, *Accepted, in press.* IEEEPress, 2009.

Following, the proposal that defines a generic method for group membership has been specified for the JXTA middleware. The specification integrates with the core services provided by JXTA in order to control peer group membership: the Membership and Access Services. Group management is achieved via additional services that take into account the idiosyncrasies of JXTA's messaging capabilities: the Patron Discovery, Sign, Trust Path Discovery and Credential Retrieval services. Furthermore, the specification seamlessly integrates with different cryptographic modules through a common interface named CryptoManager. Key authenticity is provided by defining a CBID generation process according to the JXTA peer ID syntax. The result of this work was presented in:

- J. Arnedo-Moreno and J. Herrera-Joancomartí. "Collaborative group membership and access control for JXTA". In *Proceedings of the 3rd International Conference on COMmunication System softWAre and MiddlewaRE (COMSWARE'08).*, pp. 159-166. IEEEPress, 2008.

The second part of the thesis is based on the specific JXTA framework. At this point, a complete survey of the current state of security in JXTA for basic peer operations is provided. In this survey, the whole peer life cycle is taken

into account, analyzing how the existing mechanisms interact for each of JXTA's three layers and which are the most common threats to peer operation. It is worth mention that, despite the popularity of the JXTA framework, no security survey was previously published. This research work led to the publication of:

- J. Arnedo-Moreno and J. Herrera-Joancomartí. "JXTA security in basic peer operations". *Actas de la X Reunión Española sobre Criptología y Seguridad de la Información X - RECSI*. pp. 405–414. 2008

- J. Arnedo-Moreno and J. Herrera-Joancomartí. "A survey on security in JXTA applications." In *Journal of Systems and Software, Accepted, in press.* Elsevier, 2009. ISSN: 0164-1212.

Once the state of security for JXTA has been clearly established, efforts are made towards providing additional security services at core level without the need of a specific group membership model. The proposed methods are completely based on XML security standards to guarantee peer interoperability and keep the JXTA ideary of XML protocol formatting.

On one hand, a new proposal for advertisement security in JXTA is presented, providing both advertisement privacy, integrity and authenticity in a lightweight manner, without the need of a TTP, as well as allowing control on which peers, within a peer group, can access to available resources. Its main contribution is maintaining end-to-end advertisement security for its whole lifetime, not just during transport, keeping its security layer even when stored in a peer's locally cache, until expiration. On the other hand, additional security has been provided to JXTA's core protocols, allowing privacy where it is currently unsupported, such as the multicast wire transport layer. The results of this work led to:

- J. Arnedo-Moreno and J. Herrera-Joancomartí. "Persistent interoperable security for JXTA." In *Proceedings of the 2nd International Workshop on P2P, Parallel, Grid and Internet Computing (3PGIC'08)*, pp. 354–359. IEEEPress, 2008.

- J. Arnedo-Moreno and J. Herrera-Joancomartí. "A security layer for JXTA core protocols." In *Proceedings of the 3rd International Workshop on P2P, Parallel, Grid and Internet Computing (3PGIC'09)*, pp. 463–468. IEEEPress, 2009.

- J. Arnedo-Moreno and J. Herrera-Joancomartí. "JXTA resource access control by means of advertisement encryption". *Future Generation on Computer Systems. Submitted, under revision.* ISSN: 0167-739X.

Furthermore, the results were accepted as an official SUN project:

- `https://jxta-xmlsec.dev.java.net`, requires developer account.

8.3 Further Research

There are many interesting directions to follow for further research on peer group security, be it in general, or specifically on regards to JXTA.

As far as the generic group membership model is concerned, further work includes to provide an effective method for more efficient certificate management and membership revocation mechanisms. It is interesting to study how to achieve the necessary trade-off between full peer autonomy and knowledge of the other peer's trust relationships in order to improve certificate chain search efficiency.

On regards to the JXTA specification, the Trust Path Discovery Service is still open to optimization, since it is entirely based in flooding backbone relationships, which is not efficient. It must be noted that it is not a full flooding mechanism, since requests are only propagated to trusted peers (which may be limited in number and are not necessarily equal to neighboring peers), but it could be improved in terms of efficiency.

Even though there is always room for improvement in any security mechanism, further work in enhancing peer group security in JXTA goes towards using the provided non-repudiation mechanisms in advertisements to define an incrimination protocol, allowing to inform to group members that a specific peer is publishing false information. Using collaboration mechanisms, such peer should be isolated and expelled from the group, ignoring its traffic.

Finally, it would be interesting to deploy the proposed security methods in current JXTA-based applications or frameworks, in order to provide a security layer where none currently exists. On that regard, some work has already been done for the JXTA-Overlay framework [Xha07], with the following results:

- J. Arnedo-Moreno, Keita Matsuo, Leonard Barolli and Fatos Xhafa. "A Security Framework for JXTA-Overlay". *12th International Conference on Network-Based Information Systems (NBiS'09). Accepted, in press*, 2009.

- J. Arnedo-Moreno, Keita Matsuo, Leonard Barolli and Fatos Xhafa. "A Security-aware Approach to JXTA-Overlay Primitives". *3rd International Workshop on Advanced Distributed and Parallel Network Applications. Accepted, in press*, 2009.

- J. Arnedo-Moreno, Keita Matsuo, Leonard Barolli and Fatos Xhafa. "Securing a Java P2P framework: The JXTA-Overlay case". *7th International Conference on the Principles and Practice of Programming in Java (PPPJ'09). Submitted, under review.*

Bibliography

[Alt03] J. Altman. Project JXTA: PKI security for JXTA overlay networks., 2003. `http://www.jxta.org/docs/pki-security-for-jxta.pdf`.

[AM06a] J. Arnedo-Moreno and J. Herrera-Joancomartí. Identifying different scenarios for group access control in distributed environments. *Actas de la IX Reunión Española sobre Criptología y Seguridad de la Información (IX - RECSI)*, pp. 388–399, 2006.

[AM06b] J. Arnedo-Moreno and J. Herrera-Joancomartí. Providing a collaborative mechanism for peer group access control. In *Proceedings of the Workshop on Trusted Collaboration (TrustCol'06)*, pp. 1–6. IEEEPress, 2006.

[AM07] J. Arnedo-Moreno and J. Herrera-Joancomartí. Resolución de escenarios en control de acceso a grupo en entornos distribuidos. *Actas del II Simposio sobre Seguridad Informática, Congreso Español de Informática (II - SSI, CEDI)*, pp. 119–126, 2007.

[AM08a] J. Arnedo-Moreno and J. Herrera-Joancomartí. Collaborative group membership and access control for JXTA. In *Proceedings of 3rd International Conference on COMmunication System softWAre and MiddlewaRE (COMSWARE'08)*, pp. 159–166. IEEEPress, 2008.

[AM08b] J. Arnedo-Moreno and J. Herrera-Joancomartí. JXTA security in basic peer operations. *Actas de la X Reunión Española sobre Criptología y Seguridad de la Información (X - RECSI)*, pp. 405–414, 2008.

[AM08c] J. Arnedo-Moreno and J. Herrera-Joancomartí. Persistent interoperable security for JXTA. In *Proceedings of the 2nd International Workshop on P2P, Parallel, Grid and Internet Computing (3PGIC'08)*, pp. 354–359. IEEEPress, 2008.

[AM09a] J. Arnedo-Moreno and J. Herrera-Joancomartí. Maintaining unlinkability in group based P2P environments. In *Proceedings of the 5th International Workshop on Collaborative Peer-to-Peer Systems (COPS)*, p. To be published (accepted). IEEEPress, 2009.

[AM09b] J. Arnedo-Moreno and J. Herrera-Joancomartí. A security layer for JXTA core protocols. In *Proceedings of the 3rd International Workshop on P2P, Parallel, Grid and Internet Computing (3PGIC'09)*, pp. 463–468. IEEEPress, 2009.

[Amo05] M. Amoretti, M. Bisi, F. Zanichelli, and G. Conte. Introducing secure peer groups in SP2A. In *Proceedings of the 2nd IEEE International Workshop on Hot Topics in Peer-to-Peer Systems*, pp. 62–69. 2005.

[AMss] J. Arnedo-Moreno and J. Herrera-Joancomartí. A survey on security in JXTA applications. *Journal of Systems and Software, To be published (accepted, in press).*

[Ant05] G. Antoniu, P. Hatcher, M. Jan, and D. A. Noblet. Performance evaluation of JXTA communication layers. In *Proceedings of the 5th IEEE International Symposium on Cluster Computing and the Grid (CCGrid'05)*, vol. 1, pp. 215–258. 2005.

[AOL95] AOL Inc. AOL instant messenger, 1995. `http://www.aim.com`.

[Ash97] P. Ashley. Authorization for a large heterogeneous multi-domain system. In *Proceedings of the Australian Unix and Open Systems Group National Conference*, pp. 159–169. 1997.

[Ash04] V. Ashlee. SUN's JXTA becomes big business play. *The Register*, 2004. http://www.theregister.co.uk/2004/01/30/suns_jxta_becomes_big_business.

[Aso00] N. Asokan and P. Ginzboorg. Key agreement in ad hoc networks. *Computer Communications*, vol. 23(17):pp. 1627–1637, 2000.

[Aur] T. Aura. Cryptographically generated adresses (CGA). http://www.ietf.org/rfc/rfc3972.txt.

[Bai02] D. Bailly. CBJX: Crypto-based JXTA (an internship report), 2002.

[Bal02] D. Balfanz, D. Smetters, P. Stewart, and H. Chi Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *Proceedings of the Network and Distributed System Security Symposium 2002 (NDSS '02)*. 2002.

[Bet94] T. Beth, M. Borcherding, and B. Klein. Valuation of trust in open networks. In *Proceedings of the 3rd European Symposium on Research in Computer Security (ESORICS'94)*, pp. 3–18. 1994.

[Bon07] L. Bononi and C. Tacconi. Intrusion detection for secure clustering and routing in mobile multi-hop wireless networks. *International Journal on Information Security*, vol. 6(6):pp. 379–392, 2007.

[Bri00] D. Brickling. Friend-to-friend networks., 2000. http://www.bricklin.com/f2f.htm.

[Cal01] M. Caloyannides, B. Written, and C. Landwehr. Does open source improve system security? *Software IEEE*, vol. 18(5):pp. 57–61, 2001.

[Cam99] J. Camenisch and M. Markus. Proving in zero-knowledge that a number is the product of two safe primes. In *Advances in Cryptology (EUROCRYPT'99)*, pp. 106–121. 1999.

[Cap03] S. Capkun, L. Buttyán, and J. Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing 2*, pp. 52–64, 2003.

[CCI88] CCITT. The directory authentication framework. recommendation, 1988. http://www.nist.fss.ru/hr/doc/mstd/itu/x509.htm.

[Cha91] D. Chaum and E. Van Heyst. Group signatures. In *Advances in Cryptology (EUROCRYPT'91)*, vol. 547, pp. 257–265. 1991.

[cha96] RFC 1994, 1996. `http://tools.ietf.org/html/rfc1994`.

[Che01] R. Chen and W. Yeager. White paper. Poblano: a distributed trust model for peer-to-peer networks, 2001.

[Che05] J.-C. Chen, M.-C. Jiang, and Y. wen Liu. Wireless LAN security and IEEE 802.11i. *Wireless Communications, IEEE*, vol. 12(1):pp. 27–36, 2005.

[Cla02] I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, p. 46, 2002.

[Cod] Codefarm Inc. Codefarm. `http://www.codefarm.com`.

[Coh03] B. Cohen. Incentives build robustness in bittorrent. In *Proceedings of the 1st Workshop on the Economics of Peer-to-Peer Systems*. 2003.

[Dai06] Z. Dai, Z. Fang, X. Han, F. Xu, and H. Yang. Performance evaluation of JXTA based P2P distributed computing system. In *Proceedings of the 15th International Conference on Computing (CIC'06).*, pp. 391–398. 2006.

[Dai09] L. Dai, Y. Cao, Y. Cui, and Y. Xue. On scalability of proximity-aware peer-to-peer streaming. *Computer Communications*, vol. 32:pp. 144–153, 2009.

[dc+99] DC++, 1999. `http://dcplusplus.sourceforge.net`.

[Des89] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *Advances in cryptology (CRYPTO'89)*, pp. 307–315. 1989.

[Die99] T. Dierks and C. Allen. IETF RFC 2246: The TLS protocol version 1.0, 1999. `http://www.ietf.org/rfc/rfc2246.txt`.

[Dif76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, vol. IT-22(6):pp. 644–654, 1976.

[Din01] R. Dingledine, M. Freedman, and D. Molnar. The Free Haven Project: Distributed anonymous storage service. In *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, pp. 67–95. 2001.

[Din04] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second generation onion router. In *Proceeding of the 13th USENIX Security Symposium*. 2004.

[Edi04] C. Edith, H. Ngai, and M. Lyu. Trust and clustering-based authentication services in mobile ad hoc networks. In *Proceedings of the 24th International Conference on Distributed Computing Systems Workshops (ICDCSW'04)*, pp. 769–780. 2004.

[Ell99] C. Ellison and B. Frantz. SPKI certificate theory, 1999. `http://www.ietf.org/rfc/rfc2693.txt`.

[ent] Entropy. `http://entropy.stop1984.com/en/home.html`.

[Esc02] L. Eschenauer and V. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM conference on Computer and Communications Security*, pp. 41–47. 2002.

[fas03] FastTrack, 2003. `http://developer.berlios.de/projects/gift-fasttrack`.

[Fer98] P. Ferguson and G. Huston. What is a VPN?, Cisco Systems Inc., 1998.

[FIP77] FIPS - Federal Information Processing Standard. Data encryption standard., 1977.

[Gar94] S. Garfinkel. *PGP: Pretty good privacy.* O'Reilly and Associates Inc., 1994.

[Gir91] M. Girault. Self-certified public keys. In *Advances in Cryptology (EUROCRYPT'91)*, pp. 490–497. 1991.

[gnu00] Gnutella, 2000. `http://rfc-gnutella.sourceforge.net`.

[gnu01] GNUnet, 2001. `http://www.gnu.org/software/gnunet`.

[Gok03] S. Gokhale and P. Dasgupta. Distributed authentication for peer-to-peer networks. In *Proceedings of the Symposium on Applications and the Internet Workshops (SAINT'03)*, pp. 347–353. 2003.

[Goo05] Google. GoogleTalk, 2005. `http://www.google.com/talk`.

[Gov02] D. Govoni, J. Soto, D. Brookshier, and N. Krishnan. JXTA and security. *JXTA: Java P2P Programming*, pp. 251–282, 2002.

[GP02] C. Geuer-Pollmann. XML pool encryption. In *Proceedings of the 2002 ACM workshop on XML security (XMLSEC'02)*, pp. 1–9. 2002.

[Gro04] Groove Networks Inc. Groove security architecture., 2004. Online at `http://www.groove.net/pdf/security.pdf`.

[Had02] S. Hada and M. Kudo. XML Access Control Language: Provisional authorization for XML documents, 2002. `http://www.research.ibm.com/trl/projects/xml/xss4j/docs/xacl-spec.html`.

[Hal05] E. Halepovic and R. Deters. The JXTA performance model and evaluation. *Future Generation Computer Systems*, vol. 21(8):pp. 377–390, 2005.

[Hes92] D. Hess, D. Safford, and U. Pooch. A Unix network protocol security study: Network Information Service. *Computer Communication Review*, vol. 22(5):pp. 24–28, 1992.

[Hoe04] G. Hoeper, K. amd Gong. Models of authentication in ad hoc networks and their related network properties., 2004. `http://www.iacr.org/`.

[Hub01] J. Hubaux, L. Buttyán, and S. Capkun. The quest for security in mobile ad hoc networks. In *Proceedings of the 2nd ACM int'l symposium on Mobile ad hoc networking & computing (MobiHoc'01)*, pp. 146–155. 2001.

[IEE99] IEEE. Standard specifications for wireless local area networks., 1999. `http://standards.ieee.org/wireless`.

[Jos03] E. Josefsson. IETF RFC 3548: The Base16, Base32, and Base64 data encodings, 2003. `http://www.ietf.org/rfc/rfc3548.txt`.

[jxt07] JXTA 2.5 RC1, 2007. http://download.java.net/jxta/build.

[Kal98a] B. Kaliski. PKCS#7: Cryptographic message syntax version 1.5, 1998. http://www.ietf.org/rfc/rfc2315.txt.

[Kal98b] B. Kaliski and J. Staddon. PKCS#1: RSA cryptography specifications. version 2.0, 1998.

[Kat01] J. Katz, R. Ostrovsky, and M. Yung. Efficient password-authenticated key exchange using human-memorable passwords. In *Advances in Cryptology (EUROCRYPT'01)*, pp. 475–494. 2001.

[Kau99] K. Kaukonen and R. Thayer. A stream cipher encryption algorithm (Arcfour), 1999. http://www.mozilla.org/projects/security/pki/nss/draft-kaukonen-cipher-arcfour-03.txt.

[Kaw04] L. Kawulok, K. Zielinski, and M. Jaeschke. Trusted group membership service for JXTA. In *Proceedigns of the 1st International Workshop on Active and Programmable Grids Architectures and Components*, pp. 116–121. 2004.

[Kha03] A. Khalili, J. Katz, and W. Arbaugh. Toward secure key distribution in truly ad-hoc networks. In *Symposium on Applications and the Internet Workshops (SAINT 2003)*, pp. 342–346. 2003.

[Kon01] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang. Providing robust and ubiquitous security support for mobile ad-hoc networks. In *Proceedings of the International Conference on Network Protocols (ICNP'01)*, pp. 251–260. 2001.

[Lam81] L. Lamport. Password authentication with insecure communication. *Communication of the ACM*, vol. 11:pp. 770–772, 1981.

[Lam82] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Language and Systems*, pp. 382–401, 1982.

[Lea] P. Leach, M. Mealling, and R. Salz. A Universally Unique IDentifier (UUID) URN Namespace. http://www.ietf.org/rfc/rfc4122.txt.

[Li03] Z. Li, Y. Dong, L. Zhuang, and J. Huang. Implementation of secure peer group in peer-to-peer network. In *Proceedings of Communication Technology (ICCT'03)*, pp. 192–195. 2003.

[Liu03] D. Liu and P. Ning. Location-based pairwise key establishments for static sensor networks. In *Proceedings of the 1st ACM Workshop Security of Ad Hoc and Sensor Networks (SASN'03)*, pp. 72–82. 2003.

[Luo05] J. Luo, J. Hubaux, and P. Eugster. Dictate: Distributed certification authority with probabilistic freshness for ad hoc networks. *IEEE Transactions on Dependable and Secure Computing*, vol. 2:pp. 311–323, 2005.

[Mar03] J. Marques. *LaCOLLA: una infraestructura autonoma i autoorganitzada per facilitar la col.laboracio*. Ph.D. thesis, UPC, 2003. `http://people.ac.upc.es/marques/LaCOLLA-tesiJM.pdf`.

[Met02] MetaMachin. eDonkey2000, 2002. `http://www.edonkey2000.com`.

[Mic05] Microsoft. MSN Messenger, 2005. `http://messenger.msn.com`.

[Mic07] Microsoft. MSDN library. cryptography, 2007.

[Mil88] S. Miller, C. Neuman, J. Schiller, and J. Saltzer. Kerberos authentication and authorization system. *Project Athena Technical Plan*, 1988.

[Mud03] S. Mudumbai, M. Thompson, and A. Essiari. Certificate-based authorization policy in a pki environment. *ACM Transactions on Information and System Security, (TISSEC)*, vol. 6:pp. 566–588, 2003.

[nap99] Napster, 1999. `http://www.napster.com`.

[Nik97] B. Niko and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology (EUROCRYPT'97)*, pp. 480–494. 1997.

[NIS95] NIST. FIPS PUB 180-1: Secure hash standard, 1995. `http://www.itl.nist.gov/fipspubs/fip180-1.htm`.

[NR04] C. Nita-Rotaru and N. Li. A framework for role-based access control i group communication systems. In *Proceedings of the International Workshop on Security in Parallel and Distributed Systems*, pp. 522–529. 2004.

[osp98] RFC2328: OSPF v2, 1998. `http://www.ietf.org/rfc/rfc2328.txt`.

[Pfi01] A. Pfitzmann and M. Khontopp. Anonymity, unobservability, and pseudonymity - a proposal for terminology. In *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability*, pp. 2–9. 2001.

[Pop04] B. Popescu, B. Crispo, and A. Tanenbaum. Safe and private data sharing with turtle: Friends team-up and beat the system. In *Proceedings of the Security Protocols Workshop (SPW'04)*, pp. 213–220. 2004.

[Ren08] J. Ren and L. Harn. Generalized ring signatures. *IEEE Transactions on Dependable and Secure Computing*, vol. 5(3):pp. 155–163, 2008.

[Riv01] R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Advances in Cryptology (ASIACRYPT'01)*, pp. 552–565. 2001.

[RY08] X. Ren-Yi. Survey on anonymity in unstructured peer-to-peer systems. *Journal of Computer Science and Technology*, vol. 23(4):pp. 660–671, 2008.

[Sax03] N. Saxena, G. Tsudik, and J. Hyun Yi. Experimenting with admission control in P2P. In *Proceedings of the International Workshop on Advanced Developments in Software and Systems Security (WADIS'03)*. 2003.

[Sch96] B. Schneier. *Applied cryptography*. John Wiley and Sons, 1996.

[Sha84] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology (CRYPTO'84)*, pp. 47–53. 1984.

[Sit02] E. Sit and R. Morris. Security considerations for peer-to-peer distributed hash tables. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, pp. 261–269. 2002.

[Sky02] Skype Limited Inc. Skpye, 2002. `http://www.skype.com`.

[Sta99] F. Stajano and R. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Proceedings of the 7th International Workshop on Security Protocols*, pp. 172–194. 1999.

[SUNa] SUN Microsystems Inc. Jarsigner. http://java.sun.com/j2se/5.0/docs/tooldocs/windows/jarsigner.html.

[SUNb] SUN Microsystems Inc. Keytool. http://java.sun.com/j2se/5.0/docs/tooldocs/windows/keytool.html.

[SUN01] SUN Microsystems Inc. Project JXTA, 2001. http://www.jxta.org.

[Sun03] V. Sunderam, J. Pascoe, and R. Loader. Towards a framework for collaboratiove peer groups. In *Proceedings of the 3st International Symposium on Cluster Computing and the Grid (CCGRID'03)*, p. 428. 2003.

[SUN07] SUN Microsystems Inc. JXTA v2.0 protocols specification, 2007. https://jxta-spec.dev.java.net/nonav/JXTAProtocols.html.

[Tra03a] B. Traversat, M. Abdelaziz, and E. Pouyou. Project JXTA: A loosely-consistent DHT rendezvous walker, 2003. http://research.sun.com/spotlight/misc/jxta-dht.pdf.

[Tra03b] B. Traversat, A. Arora, M. Abdelaziz, M. Duigou, C. Haywood, J. Hugly, E. Pouyoul, and B. Yeager. Project JXTA 2.0 super-peer virtual network, 2003. http://research.sun.com/spotlight/misc/jxta.pdf.

[Tra06] P. Traynor, H. Choi, G. Cao, S. Zhu, and T. La Porta. Establishing pair-wise keys in heterogeneous sensor networks. In *Proceedings of the 25th Annual IEEE Conference on Computer Communications (INFOCOM'06)*, pp. 1–12. 2006.

[Vro06] J. V. Vroonhoven. Peer-to-peer security, 2006.

[W3C02a] W3C. XML encryption syntax and processing, 2002. http://www.w3.org/TR/xmlenc-core/.

[W3C02b] W3C. XML signature syntax and processing, 2002. http://www.w3.org/TR/xmldsig-core.

[Wah97] M. Wahl, T. Howes, and S. Kille. Lightweight directory access protocol (v3), 1997. http://www.ietf.org/rfc/rfc2251.txt.

[Wal00] J. Walker. Unsafe at any key size; an analysis of the WEP encapsulation. *IEEE Document 802.11-00/362*, 2000.

[Wal03] D. S. Wallach. A survey of peer-to-peer security issues. *Software Security - Theories and Systems*, vol. 2609/2003:pp. 253–258, 2003.

[Wan07] Z. Wang, S. Das, M. Kumar, and H. Shen. An efficient update propagation algorithm for P2P systems. *Computer Communications*, vol. 30:pp. 1106–1115, 2007.

[Wei03] A. Weimerskirch and D. Westho. Identity certified authentication for ad-hoc networks. In *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks (SASN'03)*, pp. 33–40. 2003.

[Wri04] M. Wright, M. Adler, B. Levine, and C. Shields. The predecessor attack: An analysis of a threat to anonymous communications systems. *ACM Transactions in Information Systems Secururity*, vol. 7(4):pp. 489–522, 2004.

[Xha07] F. Xhafa, R. Fernandez, T. Daradoumis, L. Barolli, and S. Caballe. Improvement of JXTA protocols for supporting reliable distributed applications in P2P systems. In *Proceedings of the International Conference on Network-Based Information Systems (NBiS'07)*, pp. 345–354. 2007.

[Yea02] W. Yeager and J. Williams. Secure peer-to-peer networking: The JXTA example. *IT Professional*, vol. 4(2):pp. 53–57, 2002.

[Yea03] B. Yeager. Enterprise strength security on a JXTA P2P network. In *Proceedings of the 3rd International Conference on Peer-to-Peer Computing (P2P'03)*, pp. 2–3. IEEE Computer Society, 2003.

[Yi03] S. Yi and R. Kravets. MOCA: Mobile certificate authority for wireless ad hoc networks. In *Proceedings of the 2nd Annual PKI Research Workshop (PKI'03)*, pp. 65–79. 2003.

[Yun05] L. Yunhao, H. Jinpeng, L. Xianxian, and Z. Yu. Access control in peer-to-peer collaborative systems. In *Proceedings of the 1st International Workshop on Mobility in Peer-to-Peer Systems (MPPS'05)*, pp. 835–840. 2005.

[Zer04] K. Zerfiridis and H. Karatza. File distribution using a peer-to-peer network – a simulation study. *Journal of Systems and Software*, vol. 73:pp. 31–44, 2004.

[Zha04] L. Zhaoyu, A. Joy, and R. Thompson. A dynamic trust model for mobile ad hoc networks. In *Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems*, pp. 80–85. 2004.

[Zho99] L. Zhou and Z. Haas. Securing ad hoc networks. *IEEE Network Journal*, vol. 13:pp. 24–30, 1999.

[Zho02] L. Zhou, F. Schneider, and R. Renesse. COCA: A secure distributed online certification authority. *ACM Transactions on Computer Systems*, vol. 20(4):pp. 329–368, 2002.

[Zou05] X. Zou, B. Ramamurthy, and S. Magliveras. *Secure Groups Communications Over Data Networks*. Springer, 2005.