Succinct Arguments: Efficiency, Assumptions and Trade-offs

Alexandros Zacharakis

July 21, 2022

TESI DOCTORAL UPF / 2022
Supervisors: Vanesa Daza and Carla Ráfols
Department of Information and Communication Technologies

Thanks

I would like to thank my supervisors, Vanesa Daza and Carla Ráfols. I am grateful to both for many reasons but I would start with what I consider the most important one: you have created and maintained a healthy work environment where people are more important than results and science. As a PhD candidate, one tends to listen to a lot of "scary" stories and experiences; I am grateful I do not have such stories to share and in big part this is because of you.

Vanesa, thank you very much for the unlimited support you showed me all these years. Probably it is not evident to you, but you helped me numerous time escape my comfort zone and deal with my weaknesses. You taught me not to take the easy route, but pursue the paths that I find interesting and fascinating instead. You motivated and guided me and were always there for whatever I needed, I could not have expected more! Also working with you was inspiring in many aspects, but I want to distinguish one in particular: your passion for teaching. I wish I grow to become half as motivated, creative and didactic as you have been!

Carla, your support and motivation has been also tremendous! You have also made me prioritize the topics I find interesting and experience research as a fulfilling and fun process; you always reminded me of this when I was overwhelmed and was neglecting to have fun. I enjoyed very much our –sometimes heated– debates and I would like to thank you for perceiving me as equal in these, despite your much more experience and solid understanding. Your intuition is unique and extremely helpful, there were many moments it took a simple sentence from you to shed light to things I was struggling with for days or weeks. Finally, you always motivated me to take risky and fascinating research paths; I apologize I sometimes failed to do so.

Next, I would like to thank Matteo Campanelli, Abida Haque, Anca Nitulescu and Alessandra Scafuro for the collaborations we did.

A special thanks to Alonso González. Our collaboration has been quite a didactic experience for me. I cannot count how many times, under a lot of pressure, you chose to take the time to teach me things –from quite important to tiny details– instead of moving things faster.

I would like to thank my family for their support and my friends, who happen to be living in many different countries in this earth. Out of dire fear of forgetting somebody I would not mention you by name. I hope that we have built relations strong enough that my appreciation can be taken for granted. I hope these relations stay just as strong despite being spread around.

Anna, it would be impossible to find a way to express my gratitude in few words (or many for that matter). Instead, let me just say this: I cannot focus writing this section –the last

thing I am doing for the thesis– because I am constantly thinking my imminent moving to Berlin in exactly three days from the very moment I am typing these words! See you soon!

Special thanks to my colleagues and more importantly friends from the infamous 55.210, Abhi, Bruno, Conor, Federico, Federico, Javier, Masoud, Mohamed, Pablo, Marta, Rasoul, Sergi, Simona, Xavi and Zaira. I apologize if at times I seemed distant. As (I hope) you have understood during these years, this is far from real.

A special, well deserved thanks goes to Arantxa. I keep referring to you you as my academic sister but I would like to stop doing that now. As much as I enjoyed our collaborations, working and sharing of PhD experiences (sorry for my nagging!), it does not describe well enough our relationship. You have become one of my dearest friends and this means much more to me. I could not have been luckier in having a better companion in this journey!

For the end, let me express my appreciation to all the administrative staff (Lydia, apart from your tremendous help in so many aspects, I enjoyed very much our encounters and short discussions!) and the rest of the UPF employees, who every day put an uneven amount of effort and are -as is always the case- the untold heroes of the story.

Funding

The project that gave rise to these results received the support of a fellowship from "la Caixa" Foundation (ID 100010434). The fellowship code is LCF/BQ/DI18/11660053. This project has received funding from the European Union´s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 713673.

I received support from Protocol Labs Research Grant PL-RGP1-2021-048.

Abstract

Succinct non-interactive arguments (snarks) are cryptographic constructions that allow a prover to convince a verifier about the validity of a statement regarding some computation. We consider these objects from the perspectives of efficiency and assumptions. We modify the folding technique of Bootle et al. (Eurocrypt 16) to exponentially reduce the verifier's complexity at the expense of an updatable setup instead of a transparent one. Next, we construct a delegation scheme –which is a snark for efficiently decidable languages– using simple and well understood cryptographic assumptions. On the verification side, the construction competes in efficiency constructions that use "non-standard" assumptions. Furthermore, we consider other cryptographic constructions that are relevant to snarks. First, we explore vector commitments and consider combinatorial techniques to construct them. One of our constructions allows flexible time/memory tradeoffs. Second, we introduce folding schemes with selective verification which allows a prover to amortize the cost of producing multiple proofs addressed to different verifiers.

Resumen

Los argumentos sucintos no interactivos (snarks por sus siglas en Inglés) son construcciones criptográficas que permiten a un probador convencer un verificador sobre la validez de una declaración con respecto a algún cálculo. Consideramos estos objetos desde el punto de vista de la eficiencia y los problemas que se asumen intractables. Modificamos la técnica de plegado de Bootle et al. (Eurocrypt 16) para reducir exponencialmente la complejidad del verificador a expensas de la seguridad en generación de parámetros públicos: en lugar de ser transparentes, serán actualizables. A continuación, construimos un esquema de delegación –que es un snark para lenguajes eficientemente decidibles– usando suposiciones criptográficas simples y bien entendidas. Por el lado de la verificación, la eficiencia de nuestra construcción compite con la de aquellas que usan asunciones "no estándares". Además, consideramos otras construcciones criptográficas que son relevantes para los snarks. Primero, exploramos compromisos a vectores y consideramos técnicas combinatorias para construirlos. Una de nuestras construcciones permite concesiones flexibles entre tiempo y memoria. En segundo lugar, introducimos esquemas de plegado con verificación selectiva que le permite a un probador amortizar el costo de producir múltiples pruebas dirigidas a diferentes verificadores.

Contents

Chapter 1

Introduction

Modern cryptography has revolutionized our lives in the last half century. While it is easy to appreciate the access to information and communication means, we often omit considering how much is happening "under the hood" to make this possible. And cryptography is nothing but essential. You would not want your credit card details to be owned by the place that served you your morning coffee, a personal message you send being accessible to those having the transmission means or someone tampering with the data you receive when you search your favorite recipe on the internet. The good news is you do not need to become paranoid; cryptography to the rescue!

The common perception of "cryptography" is a way to modify messages to make them inaccessible to those who should not access them. While this is true, it is a minuscule part of a much larger and fascinating story! Webster dictionary defines cryptography as ...no, no don't leave yet, I am kidding! Cryptography is the science –or art depending on your view of the world– of *creating trust* among those who do not trust each other.

Let us see the communication example through this lens to make it more clear. You probably should not trust the communication channels you use to send a message. In the end, the message is transmitted as electromagnetic waves through wires or the air and anyone close enough to the channel can simply ...read it. Encrypting the message creates trust: it transforms it to gibberish that reveals no information about the message. For an eavesdropper, it is some useless gibberish. Similarly, when you receive information from your favorite recipe website, you cannot trust the channel. Someone can simply change the message and ruin your dinner! Again, cryptography guarantees that no one can do this. On your part, you just need to look for this green lock icon on your favorite browser. While you do not trust the channel (or the people that manage it) you created a *trusted* way of using it.

But trust is needed in many more circumstances and considering that there is almost al-

ways incentive to act maliciously, cryptography has become useful in a much broader sense. Before looking into more complex scenarios related to this thesis, let us discuss a fascinating story that lead us to where we are now: how the notion of a *proof* was revolutionized.

1.1 Revolutionizing the Notion of a Mathematical Proof

Let us start by considering the following simple question: what is a proof? Well, it depends. The intuitive meaning of a proof is some (most of the times) irrefutable evidence that asserts a claim. But there are a lot of things to consider, especially in the context of mathematics.

First, we need to discuss what a claim is. A claim is simply a statement of a theorem[1] in a well defined mathematical framework. Until very recently, a proof of a theorem has been considered to be a *static object*. A specific application of some rules in some order that leads from some axiomatic truth to the claim in question. One simply needs to write down the rules and the recipient of the proof –the *verifier*– has simply to read them and accept it or reject by essentially verifying the rules are correctly applied. In the case that the two parties agree on the axiomatic statements of the system, a proof is indeed irrefutable evidence of the validity of the claim.

In hindsight, however, one can argue that this is a very limiting setting. Our end goal is in fact to assert validity of a claim and maybe there are easier ways to do this. Let us motivate ourselves by recalling a quote of Shimon Even. In his personal website, Oded Goldreich shares the following story: *"In 1978, as an undergraduate, I attended Shimon's course Graph Algorithms. At some point, one student was annoyed at Shimon's "untraditional" way of analyzing algorithms, and asked whether Shimon's demonstrations constituted a proof and if so what is a proof. Shimon answer was immediate, short, and clear: A proof is whatever convinces me"* [Gol11].

Defining a proof this way, it is enough to consider ways that we use to convince ourselves about the validity of statements. One strategy to avoid "writing down" a proof we use in our everyday lives' is *interaction*. Simply ask questions to whomever makes a claim and see if they can give convincing answers! Interaction is a natural way of extracting truth. And it has to have some benefits over statically produced proofs; in the end, it is considered universally harder to understand a manuscript than discussing with a "teacher".

The second ingredient to redefine "proofs" is less intuitive: *randomness*. The person we "interrogate" might be clever enough to predict our questions and prepare to give ap-

[1]This is equivalent to considering the output of a computer program. The "theorem", for example, could be "The shortest path from Paris to Madrid is no more than 1300km". This notion is more meaningful in practice, but we prefer the example of traditional proofs of theorems for this discussion.

propriate answers that will convince us about false claims. We need to add some unpredictability to the equation, and to do that, we simply use a coin. We flip the coin, and our questions depend on the coin flips. Hopefully, it is harder to be fooled this way!

Indeed, these two ingredients –interaction and randomness– have changed our perception of what a proof is. And if we are willing to accept some minuscule error, meaning that we might be convinced about a false statement but only with extremely low probability –for example 2^{-100}– we can indeed do much cooler things than simply considering static proofs. The introduction of such proofs, from now on *interactive proofs*, was done by Goldwasser, Micali and Rackoff [GMR85] and independently by Babai [Bab85]. It has been one of the most important and influential results of Theoretical Computer Science. These works did not only challenged the classical notion of a mathematical proof, but also inspired and continue to inspire generations of researchers.

Shortly after these works were published, one of the most important and counter intuitive properties was introduced: *Zero Knowledge Proofs* [GMR85]. As the name suggests, these proofs convey no knowledge from the prover to the verifier, or to be more accurate, no knowledge except the validity of the statement. This means that somebody can make a statement and convince us that it is true *without revealing anything else about it*. It simply conveys a single bit of information: truth or false! If this sounds like magic it is probably because it (almost) is!

Consider the implications of such a discovery. The *privacy* of a prover is preserved while it still manages to convince the verifier. And this is how trust is born. Two parties that do not trust each other, interact and flip coins, and at some point the verifier is convinced about some statement and the prover is guaranteed to not have leaked any sensitive information during the process. While these were mainly objects of theoretical interest, recently they have been deployed in many real world applications and tremendous efforts have taken place to understand and improve them.

In a parallel line of work, people considered a different but related question regarding the *verification cost* of a (classical) proof. The question is simple to conceptualize: should we read the proof as a whole to convince ourselves about the validity of a claim? While the natural answer seems to be positive (if not need to read the whole proof, why even include the unnecessary parts?) things change when we allow randomness. We ask the prover to modify the proof by adding some redundant parts (this makes the proof longer but not by much in an asymptotic sense) and we do a few spot-checks. Intuitively, the malicious prover does not know which part of the proofs we will check a priori.

This idea led to the introduction of *Probabilistically Checkable Proofs* where the verifier has a static proof, chooses some random locations, reads them and is ready to decide whether to accept or reject. The celebrated PCP Theorem [AS92] states that for all **NP** statement (and thus all the proofs we can hope to produce) it is enough to flip *logarithmic coins* in the size of the statement and just see a *constant number of proof elements*!

Not long after that, the idea of *succinctness* came into consideration. The goal was to return to our initial point: proofs as static objects. But as the name suggests, these proofs have an important property: they are very small is size, asymptotically smaller than a proof in the traditionally sense. The reader at this point might be wondering how is this even possible. A proof cannot be "compressed", this would just constitute a more laconic proof. The answer is indeed that we can not do that unconditionally. But we can circumvent this!

The fact we exploit is that we live in a world were resources are limited. A cheating prover trying to convince us about a false statement can only perform *efficient computation*. Therefore, it does not matter if there are convincing proofs *as long as it is infeasible to compute one of them!*

And this is the point where everything connects. We take the heavy machinery of IPs and PCPs and use some *computational assumptions* to achieve succinctness. We assume that some problems are computationally hard[2] and we create protocols that are secure if our assumptions hold. In the context of proof systems it is guaranteed that there exist proofs of false statements, but it is computationally hard to find them. The notion of soundness of mathematical logic now becomes *computational soundness*.

Succinct proof systems are related, directly or indirectly, to all the chapter of this thesis. We give a high-level overview of such objects in the next section.

1.2 Succinct Non-Interactive Proofs

This section aims to explain what succinct non-interactive proof systems are. We intentionally keep the discussion informal; the goal is to give motivation and intuition about these objects and discuss the different perspectives to look at them.

First let us explain what is the problem we are trying to solve. As we mentioned, we have a prover that wants to convince a verifier about a statement. Lets formalize it a bit. We consider a set $\mathcal{L}$ defined by a predicate $\mathcal{R}$ as follows:

$$\mathcal{L} = \{x \mid \exists w \text{ s.t } \mathcal{R}(x, w) = 1\}$$

We call $\mathcal{L}$ the language. This is simply a set that consists of all *statements* x for which there exists a *witness* w that makes the *relation* $\mathcal{R}$ corresponding to $\mathcal{L}$ true. We also require that predicate $\mathcal{R}$ is efficiently computable, that is, there exists a polynomial (in the size of x) algorithm that computes it. For example, $\mathcal{L}$ could be the language of Hamiltonian graphs: x is the encoding of a graph, w the Hamiltonian cycle and $\mathcal{R}$ the predicate that checks if w is indeed a Hamiltonian cycle of the graph. Note that the predicate is indeed efficiently

[2]We can only assume because we do not know how to actually prove this fact or if it is even truth for that matter. We are far from understanding computation and its properties in a mathematical sense.

computable: simply assert that all nodes of the graph are included in the claimed cycle and that there is an edge between each consecutive pair in the claimed cycle.

Let us now consider the prover and the verifier and our first (trivial) protocol. The prover claims that $x \in \mathcal{L}$. There is a simple way to convince the verifier about this statement. Simply send w. The verifier will then check if $\mathcal{R}(x, w)$ is accepting or not. We can easily argue about its properties:

1. The protocol is *complete*. As long as the prover knows x it will always convince the verifier by definition of $\mathcal{L}$.

2. The protocol is *sound*. If $x \notin \mathcal{L}$, no matter what the prover sends, the verifier will always reject, again by definition of $\mathcal{L}$.

For those familiar with complexity theory, this is, in fact, one of the formulations of the complexity class **NP** consisting of languages decidable in non-deterministic polynomial time. This language captures all the computation that can be *verified efficiently*.

The "trivial" protocol works, but there are two things we would like to improve: (1) reduce communication, that is sending something smaller than w, and (2) protect the privacy of the prover: the witness w might be valuable and the prover unwilling to share it despite the fact that it wants to convince the verifier about $x \in \mathcal{L}$. For example, when the statement is "ciphertext c decrypts to m" you would not want to send your secret key that could decrypt a bunch of other messages!

The former notion is called *succinctness* and the latter *zero knowledge*. Succinctness is almost universally desired when considering application while zero knowledge is optional, depending on the application. In our setting, it is the former that complicates things. Usually, a succinct proof system can be "easily" augmented with zero knowledge.

Let us now be a bit more concrete and define these notions. To do that, we will make some relaxations. First, we allow the parties to have access to some parameters we call the *structure reference string*. We assume that some trusted party magically gives this to our parties, although a lot of discussion can be done on the matter. The reason we consider such parameters is simple: it is unavoidable[3]. Second, we will relax the soundness property. We will restrict the (possibly malicious) prover to be a probabilistic polynomial algorithm. This means that it is allowed to flip coins and it is restricted to perform an (a priori fixed) polynomial number of steps in the size of its input. This is universally considered to capture the notion of *efficient computation*.

Let us now introduce the notion of a succinct non-interactive argument. Such an argument consists of two efficient algorithms Prove and Verify that work as follows:

[3]We can however circumvent the restriction we posed about the honest generation of them.

Prove: takes as input the structure reference string srs, the statement x and a witness w and produces a proof π.

Verify: takes as input the structure reference string srs, the statement x and a proof π and outputs true or false, representing if the proof is correct or not

Let us now see the basic properties we would expect, keeping in mind the relaxation we did about the prover:

1. *Completeness*: for all $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$, algorithm Prove will produce an accepting proof.
2. *Computational Soundness*: for all $(\mathsf{x}, \mathsf{w}) \notin \mathcal{R}$, no *efficient* algorithm can produce a proof π^* that will make $\mathsf{Verify}(\mathsf{srs}, \mathsf{x}, \pi^*)$ output true.
3. *Succinctness*: the size of the proof π is sublinear in the size of w, namely $|\pi| = o(|\mathsf{w}|)$.

Now let us give a stronger definition of computational soundness. We want to capture the following notion: not only there exists a witness, but the prover *knows* one. Let us motivate this with an example. Consider the following language:

$$\mathcal{L} = \left\{\mathbb{G}, g, h \mid \exists x \text{ s.t. } \mathbb{G} \text{ encodes a cyclic group of order } q \text{ generated by } g \text{ and } h = g^x\right\}$$

Finding a witness x is known as the *discrete logarithm problem*. For many groups (or more precisely group families of increasing size) the problem is assumed to be computationally hard: there is no polynomial algorithm that can solve it. However, deciding membership in $\mathbb{G}$ is efficient. Therefore, the language is in fact efficiently decidable. Simply check that $h \in \mathbb{G}$, and by the fact that the group is cyclic, such a witness *must exist*. Someone claiming *it knows* such a witness for x makes a stronger claim.

We formulate this by considering a knowledge extractor. We emphasize that this notion is non-trivial since algorithms do not "know" things. So how do we model this? Well, an algorithm "knows" the stuff that it can compute using its state. For example, if an algorithm $\mathcal{A}$ with input x computes $h = g^x$, we can simply "modify" this algorithm to a new algorithm $\mathcal{A}'$ that outputs $g^{x^{42}}$. So in some sense, $\mathcal{A}$ "knows" $g^{x^{42}}$. A bit more abstractly, an algorithm $\mathcal{A}$ knows something, if there exists another algorithm $\mathcal{E}$ that knows how $\mathcal{A}$ works and can output this value.

We are now ready to express the stronger property:

4. *Computational Knowledge Soundness*: for all efficient $\mathcal{A}$ that output (x, π) that passes the verification, there exists an efficient algorithm $\mathcal{E}$ –called the knowledge extractor– that outputs w such that $\mathcal{R}(\mathsf{x}, \mathsf{w})$ holds.

Note that the order of quantifiers ($\forall\mathcal{A}\ \exists\mathcal{E}$) means our extractor can depend on $\mathcal{A}$ and therefore can know how it works:[4] simply consider the extractor that has $\mathcal{A}$ hardcoded in

[4]There are various meaningful extraction definitions and in fact, properly defining it is more complicated

its description. Also, note that this property implies the normal computational soundness: if $\mathcal{E}$ can output a valid witness, such a witness must exist in the first place.

A construction satisfying these properties is known as a Succinct Non-Interactive Argument (SNARG) if it simply satisfies computational soundness or Succinct Non-Interactive Argument of Knowledge (SNARK) if it is satisfies computational knowledge soundness. The first such construction was due to the seminal work of Micali [Mic94] who built on the work of Kilian [Kil92].

The discussion so far tries to protect the verifier from a malicious prover. But what happens if the verifier is malicious? First, let us give some context for the discussion. A verifier is supposed to only learn a single bit of information: whether $\mathsf{x} \in \mathcal{L}$ or not. Ideally, it should learn nothing more because it simply "happens" to engage with the prover to a protocol. As motivation, consider the discrete logarithm example we discussed before. The fact that the prover knows a value x such that $h = g^x$ can be used to identify him. Since the problem is hard, the prover must have computed h by sampling x and doing the exponentiation. So x can be considered the secret key and h the public key he shared with the world. No one can learn the secret from h since computing discrete logarithms is (assumed to be) hard! In this scenario, when identifying the prover, you would not want the verifier to learn information about the secret key.

This property is called *zero knowledge* and it captures precisely the fact that the verifier learned nothing but the validity of the statement. In the author's personal view, the conception and formalization of this notion is one of the greatest intellectual achievements of humanity.

What we require informally is that we can efficiently produce a tuple $(\mathsf{srs}, \mathsf{x}, \pi)$ that looks the same as getting an srs and executing $\pi \leftarrow \mathsf{Prove}(\mathsf{srs}, \mathsf{x}, \mathsf{w})$ for a valid statement/witness pair. At this point, one might wonder how this does not contradict soundness; the answer lies in the fact that in the former case we are allowed to sample srs ourselves. The fact that we can produce a tuple $(\mathsf{srs}, \mathsf{x}, \pi)$ that looks identical with one that comes from honestly proving a statement *without the witness*, means that we learn *no new information* about the witness. Indeed, if we learned something during the interaction with the prover, we could avoid interacting with the prover in the first place and simply produce the tuple that was supposed to produce new knowledge ourselves! Thus, after seeing the proof, we know exactly what we knew before! Note that the argument (kind of) describes extracting knowledge from the malicious verifier.

Things are more complicated in practice. For example, we can actually do something we could not do before: produce a proof about the statement x. We simply hand the fancy proof we received! Still, on a philosophical level, we did not learn something about the witness w, but rather we learned something about the specific srs: how an accepting proof for x looks *with respect to it*.

than what is presented in this section.

Let us now formulate the notion.

5. *Zero Knowledge*: Let $\mathcal{D}$ be the distribution from which srs is sampled and (x, w) any statement/witness pair that satisfies the relation $\mathcal{R}$. A SNARG (resp. SNARK) is zero knowledge if there exists an efficient algorithm $\mathcal{S}$ –the simulator– that outputs $(\mathsf{srs}, \mathsf{x}, \pi)$ that are *identically distributed*[5] to first sampling $\mathsf{srs} \leftarrow \mathcal{D}$ and then running $\pi \leftarrow \mathsf{Prove}(\mathsf{srs}, \mathsf{x}, \mathsf{w})$.

We emphasize that the most difficult thing to achieve is computational soundness assuming succinct proofs. Usually, in this setting, zero knowledge comes (almost) for free. There is an intuitive interpretation for this fact: a succinct proof is so small that it cannot contain a lot of information about the witness. But compressing to that extent the information contained in the witness in a way that it remains convincing is really hard. This is precisely the reason for achieving only computational soundness. We inherently need to assume that certain computations are infeasible. In fact, we *need to make quite strong assumptions*, probably stronger than what our confidence in understanding computation should allow. This is usually justified, however, by impossibility results.

1.3 Efficiency Requirements

It should be clear by now that one (perhaps the most important) goal is to have constructions that are *efficient*. In the end, succinctness aims to having proofs that are as small as possible. Understanding efficiency in the setting of succinct arguments is first of all a matter of *theoretical interest*: what is the smallest proof we can send that still convinces a verifier? But since these constructions are actually now deployed and used in real world applications, efficiency becomes also of *practical importance*.

After conceptualizing the notion of a succinct argument, a natural question to ask is "*how succinct can a succinct proof actually be?*". And –as usual– there is no single satisfying answer. You could probably make a snark with smaller proof size but you might need to use stronger assumptions and/or make efficiency worse in some other aspect. In this section we discuss these various aspects. In the next one, we will consider the assumption aspect of snarks.

Before discussing these aspects, we emphasize that there exists a soundness error: we accept that a verifier will be convinced about a false statement with small probability. This notion is captured by the security parameter: this is a number $\kappa \in \mathbb{N}$ that defines what this error will be and we can make it arbitrarily small by considering the appropriate security parameter. All efficiency measures depend on the size of κ: higher security parameter translates to smaller error and less efficiency. For simplicity, we omit this in the discussion that follows but we emphasize that we always pay some price related to κ.

[5] As with the case of knowledge soundness, there are various other flavors one could consider.

Let us start from the obvious efficiency aspects. First, the verification time should be minimized. In the end, the final goal is to be able to assert claims efficiently and succinct proofs is simply a necessity for that: you need to read the proof so the running time of the verifier is lower bounded by it (recall we are in the setting of static proofs, we do not consider probabilistic checking)[6]. Ideally, you want the verifier time to be linear to the proof size. Conceptually, the verifier should not do much more work than reading the proof!

Second, the prover should be fast. Of course, creating a proof involves reading and processing the witness so there is always some overhead. The goal is to minimize this overhead. The measure to compare here is the *time to assert the statement in the classical sense*, that is, the number of computational steps to compute $\mathcal{R}(x, w)$. Note that this actually defines the practicality of the construction: having a slow verifier might be undesirable but a slow (for example quadratic) prover would make the computation infeasible in practice for averaged size computations.

Finally, we mentioned that the prover and verifier share a common reference string. Until now, we assumed that this just magically appeared, but this also should be computed. Therefore, we require that (1) its size is small and (2) it is efficiently computable. There is also another matter of efficiency regarding this. Since this string involves trust in many snark constructions, a simple computation is usually not enough. Parties need to engage in complex multi-party computation protocols to come up with it which can be really difficult to realize in the real world. As we will see in the next section, effort is given to mitigate this as much as possible.

While these can be considered universally applicable measures, one could also consider amortized costs. In general, snarks are instantiated and are intended to be used to prove various statements in a long lifetime. Amortization considers the efficiency of proving or verifying k statements. Having for example two statement/proof pairs $(x_1, \pi_1), (x_2, \pi_2)$ it would be useful to be able to combine the two proofs to a single proof π^* that is convincing for both statements. To be non-trivial, proving/verifying k statements with a single proof should be smaller than doing the same process k times independently. Amortization techniques are mainly used to compensate for a slow verifier. Such a verifier might not be an issue if it does the "heavy work" once for many proofs. Similarly, since constructing a proof has an overhead, a way of proving many statement while paying this overhead once can be beneficial in practice.

[6]We note however that even succinct constructions without succinct verification exist and can be useful in practice since they reduce *communication* which is the bottleneck in various applications.

1.4 Computational and Trust Assumptions

As we mentioned when introducing snarks, we need to settle for computational soundness: convincing proofs for false statements exist but it should be infeasible to construct one. This inherently requires that some problems are computationally hard[7]. Unfortunately, we do not know of any (suitable) problem whose hardness can be proven and it is actually debatable whether we are even close to such a discovery.

Nevertheless, there is an astonishing large number of problems we are unable to solve *despite of the fact that we have put tremendous efforts to this goal*. Perhaps the most notorious example is factoring big numbers[8]. The problem is so elegant and natural. Given a number n that is the product of two large primes, i.e. $n = p \cdot q$ with p, q having κ digits, find p and q. Mathematicians have tried for literally hundreds of years to solve the problem with no success.

Our inability to solve some problems combined with our inability to prove their hardness leads to the expected result: make a conjecture that the problem is hard. These conjectures are called computational assumptions and are extensively used in cryptography. Usually, a statement about the security of a cryptographic construction becomes of the form "if problem X is hard then construction Π is secure". To prove such a statement, you do something quite elegant: you show that if you can indeed break construction Π, you come up with an algorithm that solves problem X. Now, if the latter is indeed hard, then obviously your construction is secure! We can even characterize the situation as a win-win: either we have fancy cryptographic tools or we have efficient algorithms for problems we did not even dream to exist.

There is an obvious consequence though. The trust we created using a cryptographic solution is in its core trust in the assumption we use. If you believe that factoring is easy, you should not use the RSA cryptosystem that bases its security on it. But the next natural question is "what assumptions should we should use" and "what do we do when we cannot construct some specific primitive under the assumptions we normally use"?

The case of snarks. As we mentioned earlier, snarks are a very powerful tool. Succinctness is too strong of a requirement and therefore constructing snarks is inherently complex. The core of the problem is that we need to "compress" a witness to the extreme. Complex in this context can be interpreted as "technically involved", but it also has another meaning: we need very strong tools (i.e. assumptions) to construct them. A result due to Gentry and Wichs [GW11] ties our hands: snarks *do not exist* under the assumptions we normally use to build, for example, encryption or signature schemes. We are left with two options: abandon our goal or use stronger, "non-standard" assumptions. The temptation is high, so –as you may have guessed– we go for option two. Apart from computational assumptions, though, snarks might require some trust assumptions. Specifically,

[7]Note that the definition of computational soundness itself for a fixed construction is such a problem.
[8]This is actually an easy (in a complexity sense) problem when one considers quantum computers.

it might be the case that the structured reference string is sampled in a way that requires some trust and we need to mitigate this issue as much as possible. Next, we consider the complexity and trust assumptions used in snarks.

1.4.1 Classifying Complexity Assumptions

Complexity assumptions are statements about the computational hardness of problems. As we said, a standard way to assess the validity of an assumption is to try to attack it: find an algorithm to solve the problem in question. The more we fail, the stronger we should believe the assumption is true.

But we have also other ways to argue about the hardness of assumptions: relate the difficulty of solving a problem X to that of solving another problem Y. If we show that solving X implies that we can also solve Y, then we can trust hardness of X as long as we trust hardness of Y! The assumption "X is hard" is stronger than the assumption "Y is hard". This approach is inspired by the study of **NP** completeness (and computational complexity as a whole) and has been proven very useful in assessing the relative difficulty of computational problems.

This has led to *assumption families* of increasing difficulty. Let us consider an example of such a family. We already discussed the discrete logarithm problem: given $g, h \in \mathbb{G}$ find x such that $h = g^x$. We can now consider a family of increasingly stronger assumptions as follows:

$$\text{"Given } h_0 = g^{x^0}, h_1 = g^{x^1}, \ldots, h_q = g^{x^q} \text{ find } x\text{"}$$

For each $q \in \mathbb{N}$ we have a different assumption. Furthermore, setting $q = 1$ we get the discrete logarithm assumption. We next make a simple observation: the larger the q, the stronger the assumption becomes. Indeed, consider two assumptions for $q_1 < q_2$ and assume you can break the assumption for q_1. Then obviously you can solve the assumption for q_2: simply ignore the elements h_i for $i > q_1$. We have also evidence that for larger q, the assumption becomes strictly stronger [BFL20].

Obviously, we should try to make constructions with q as small as possible. But it is also important to *separate* the cryptographic construction from the assumption used. For example consider a snark construction that is secure under a q-type assumption, as long as q is at least as large as the description of the computation we want to prove. It could be the case that for some languages the snark is secure and others not. In general, we want to use assumptions that are simple and independent of the construction we are proving secure. This also connects with the notion of *falsifiability* which we discuss next.

Assumptions have been classified as falsifiable and non-falsifiable. This classification was introduced by Naor [Nao03], but a reformulation due to Gentry and Wichs [GW11] is currently most widely used. The notion we try to capture with falsifiability is the win-win situation we described earlier: either the assumption holds or we solve a previously

unsolved problem.

The notion is captured as follows: an assumption is falsifiable if we can interact with a challenger and at the end of the interaction we can efficiently decide if the challenger breaks the assumption or not. The discrete logarithm is such a case: sample h randomly, send it to the challenger, receive x and check if $h = g^x$ or not. The latter test is efficient, so we know if the challenger succeeds or not.

Let us contrast this with a non-falsifiable assumption to make the distinction clearer. We will consider the "knowledge of exponent" assumption introduced in [Dam92]: "any efficient algorithm $\mathcal{A}$ that on input $g, h \in \mathbb{G}$ outputs g^z, h^z must have first computed z and use that to produce the result". In this case, we cannot know if a challenger broke the assumption by simply interacting with him.

A crucial difference between the two assumptions is that the former considers *what* an efficient algorithm can do while the latter *how* it does it. This is in tension with the very reasons that gave rise to modern cryptography. Before not too long ago, we would construct an encryption scheme using some "clever" way and if we could not break it ourselves, we would consider it secure. This implicitly makes a strong and unjustifiable assumption: the only viable strategies (and hence the ones an adversary would use) are the one we tried. Modern cryptography revolutionized such constructions by considering what an adversary does instead of how it does it: no matter what strategy an eavesdropper uses, it cannot break the scheme as long as a computational problem is hard.

Another approach to constructing cryptographic tools in general and snark in particular, is restrict our attention to some *idealized* scenarios. We observe something in the real world, abstract it and only consider adversaries that "live" in this abstractions. The most notable examples are the random oracle model [FS87] and idealized models regarding groups [BL96; Sho97; FKL18].

The random oracle model assumes that an adversary is using a hash function as a black box, i.e. without access to how it works. Obviously, this is not the case in the real world. We rely on the fact that we cannot exploit the inner workings of hash functions to break some cryptographic primitive. The idealized group models make assumptions related on how the adversary uses group elements on constructions that rely on groups. The common denominator in these models is, in some sense, that the adversary can only produce new group elements using the group operations.

While results in such models have advanced the field, they should probably not be considered the end goal, but rather steps towards better understanding that will eventually lead to better and more secure constructions. In fact, in the case of the random oracle model, we know that it is not secure theoretically [CGH04; GK03], but there are no known attacks on practical constructions. The insecurity is proven by considering specially crafted constructions designed to make the model fail.

In the case of snarks, where we know that we cannot base security on falsifiable assumptions, things are more complex. We are inherently limited to using non-falsifiable assumptions, but it is probably a good idea to not rest at ease. We should still try to minimize the use of such assumptions and only use them when they are unavoidable. In the end, these strong assumptions are objects we do not fully understand and we have evidence to doubt their security.

A relevant reference for the reader interested to look into the quality of assumptions in more depth is [GK16].

1.4.2 Trust Assumptions

As mentioned before, a snark construction needs a *structured reference string* (srs). This is some string that is produced once in an offline phase, that can be then used to prove arbitrarily many statements. And as we mentioned, in some cases we need to assume that this string is honestly generated. Failing to satisfy this requirement can translate to the ability of producing proofs of false statements, making the construction effectively useless.

Assume a snark construction needs a trusted srs. This essentially means that if we know the randomness used to sample it, construction of false proofs is feasible. We have two options. First, we can try to find a party that is universally trusted and assign it with producing it (and deleting any information that could compromise the system, the so called "toxic waste"). This would work perfectly but of course such parties are not easy to find... We do the next best thing, trying to mitigate the issue.

The second option is to find many parties and assign them the collective creation of the srs. This is done through complex cryptographic protocols that give the following guarantee: if even one of the parties behaves honestly, we are guaranteed that the parameters are trusted. Thus, we reduced the trust problem to trust one out of many parties instead of trusting a single one, which is of course better. However, this process is extremely complex and difficult to execute correctly to get the last guarantee. Perhaps the most famous example of executing such process is the generation of the srs for the ZCash cryptocurrency that utilizes snarks [ZCa21], which was extremely involved.

To make things even worse, for many snark constructions the setup parameters supports a specific language $\mathcal{L}$. Thus, if you want to change the language (e.g. consider an update of a deployed application) you need to execute this process from scratch. And even if everything is done correctly, you still have to trust that the participants did not collude.

A line of work focused on improving the situation. Groth et. al. [GKM+18] introduced the notion of *universal* and *updatable* snarks. Universal means that the parameters can be used

for instantiating any computation[9] without the efficiency overheads incurred by considering universal computations. This already solves the latter issue: the ceremonies for the srs need to be performed once. The updatable part of the notion refers to a specific type of multiparty computation needed to construct the parameters: an unbounded number of participants can participate in a non-interactive way. Essentially, we start with some parameters srs_0, a participant *updates* them to srs_1, another participant to srs_2 and so on. At any point you can use the parameters. Thus, a very larger amount of people can participate in the creation of the srs and the trust assumption is much weaker. In fact, each user of the system can choose to participate and she can trust the parameters if she trusts herself!

There is also a third solution that is the optimal. The parameters are produced by sampling some uniform element from some set. Snarks that use such parameters are called *transparent*. Assume we have a way to sample a uniformly distributed seed ρ for the parameters. Then, each user can simply compute srs from a deterministic procedure that takes as input ρ. The trust assumption is now much easier to enforce: it is enough to know that ρ was indeed sampled uniformly. This is an easy problem: we rely on the unpredictability of the natural world. The drawback of transparent constructions is that they are in general less efficient. The reason for that is that the srs is uniform and thus carries no information. In contrast, in the previous cases we have an srs that is highly structured and we exploit this very structure in our constructions.

1.5 Our Results

In this section, we take a brief look at the results that will be presented in this thesis.

Improved Inner Product Argument

Our first result, presented in Chapter 3, is a modification of the widely used protocol of Bootle et. al. [BCC+16]. The protocol is known as the "folding technique" and has various applications. In its core, it allows to convince a verifier that the openings of two (short) commitments/hashes are vectors whose inner product is z. The protocol has wide applicability. The inner product relation is powerful enough to imply various cryptographic tools, including snarks. It also has very appealing properties: the proof size is *logarithmic* in the size of the vectors and it does not require a trusted setup. On the downside, the verification cost is linear in the size of the vectors.

We reduce the verification cost *exponentially*, making the verifier logarithmic in the size of the statement. The key idea to achieve this is considering generalizations of the classical Pedersen commitment scheme. Specifically, instead of considering a uniformly dis-

[9] To be accurate, any computation whose size is bounded by some parameter used to generate the srs.

tributed key, we sample it pseudorandomly. This allows us to have a succinct representation of it (the seed used for sampling). Working in the pairing group setting, we manage to exploit this succinct representation to reduce the work of the verifier. In doing so, however, we no longer have a transparent setup. Nevertheless, basing the commitment key to monomial-based distributions, the trusted setup is the weakest possible: it is updatable. This means essentially that there exists a non-interactive protocol to construct parameters that allows participation to an unbounded number of parties.

Delegation of Computation

A delegation scheme is essentially a snark for efficiently computable languages. This means that we can decide if $x \in \mathcal{L}$ ourselves, but we ask another party to convince us about the fact because we are not willing to perform this computation. Consider for example asking a cloud server to perform a heavy computation on your behalf. While snarks for languages that are not efficiently decidable require non-falsifiable assumptions or idealized models, this is not the case in delegation schemes. In fact there exist constructions that achieve this goal under falsifiable assumptions.

In Chapter 4 we contribute in this direction. We construct a delegation scheme under standard, constant size assumptions. In fact, the assumptions we use are quite simple and "standard" in the sense that they are already extensively considered in the literature. Furthermore, we manage to achieve *constant proof size* and verification overhead. As far as verification is concerned, the construction competes with snarks that use strong knowledge assumptions or the algebraic group model. The prover, however, is quadratic which makes the construction impractical.

This result is evidence that in the case of delegation of computation, we could potentially avoid using any non-standard assumption and simply rely on the same assumptions we use to build encryption and signature schemes. Furthermore, the techniques we use combine very interesting techniques from the perspective of cryptography and computational complexity which –to the best of our knowledge– were not combined before.

Tree Based Vector Commitments

Next, in Chapter 5 we explore combinatorial techniques for a cryptographic primitive called *vector commitments*. This primitive allows to *commit* to a vector of values and then decommitting to any position. They have practical applications per se and they are a core ingredient in a family of snark constructions. The most widely known example of such constructions are Merkle trees.

First, we consider the problem of constructing such a primitive in the discrete logarithm setting. We first construct generic constructions using only combinatorial and algebraic

properties. We then instantiate the construction using the folding technique to get our commitment scheme. The construction inherits the properties of the folding technique; notably it has a transparent setup. Furthermore, the combinatorial and algebraic structure gives additional properties, notably homomorphicity and efficient proof pre-computation.

The second construction is a maintainable vector commitment construction in the pairing group setting that supports various trade-offs. Maintainability essentially captures that the amortized cost of proving statements about the vector is small in applications where the vector regularly changes over time. Our contribution in this direction is to improve the state of the art construction by reducing the proof size by a constant factor. Furthermore, we introduce new trade-offs that are important in practice: we allow a prover to trade memory resources for computation resources in an arbitrary way. This results in the following trade-off: more memory resources means less computation time per opening and larger proof size. Depending on the application and the available resources, one can choose what memory and time requirements it should settle for.

Folding Schemes with Selective Verification

Finally, in Chapter 6 we present a novel construction that allows a prover to convince multiple verifiers about multiple independent statements. Specifically, the prover can compute a single proof for a "batch" of statements and later convince a verifier about the validity of one of the statements. Importantly, the size of the proof is *sublinear* in the total number of proven statements.

Since, in general, aggregating statements is efficient while proving them needs a lot of work, such a construction can be beneficial in cases where a single prover needs to convince multiple verifiers: instead of producing one proof for each, it can aggregate the statements and do the work once. The verifiers' overhead is minimal and, furthermore, they do not need to know what were the other statements considered. In particular, it requires no communication among the verifiers. Finally, this is a novel concept which seems interesting per se. We believe it has the potential of further practical uses.

Chapter 2

Preliminaries

2.1 Notation

We denote the set of natural numbers by $\mathbb{N}$ and let $\kappa \in \mathbb{N}$ be the computational security parameter. We denote with $[n] = \{1, \ldots, n\}$ the set containing all natural numbers not greater than n.

When $\mathcal{D}$ is a distribution, we denote with $x \leftarrow \mathcal{D}$ a value x sampled according to $\mathcal{D}$. For a set S, we write $x \leftarrow S$ to denote uniformly sampling from S and assigning to x. Similarly, when A is an algorithm we denote with $y \leftarrow A(x)$ the assignment of the output of A with input x to y, where we uniformly sample randomness from A if it is probabilistic. We write $A(x; r)$ to explicitly refer to the randomness of A when needed. We notate with $O(\cdot)$ asymptotic complexity that hides linear factors that depend on the security parameter κ. Unless otherwise stated, all the algorithms defined throughout this work are assumed to be probabilistic Turing machines that run in polynomial time (abbreviated as PPT). We say that a function is *negligible* (in κ), and we denote it by negl, if $\mathsf{negl} = \Omega(\kappa^{-c})$ for any fixed constant $c > 1$.

Vectors and Matrices. Let $n \in \mathbb{N}$ and $\mathbb{F} = \mathbb{Z}_p$ for a prime number p. We denote with $\mathbf{e}_i^n \in \mathbb{F}^n$ the i-th element of the canonical basis of $\mathbb{F}^n$, that is $\mathbf{e}_i^n$ is 1 in the i-th coordinate and 0 everywhere else. We omit the superscript n when it is clear from the context.

For vectors $\mathbf{a} = (a_i)_{i\in[n]}, \mathbf{b} = (b_i)_{i\in[n]} \in \mathbb{F}^n$, we denote

- $\mathbf{a}^\top \mathbf{b} = \sum_{i=1}^n a_i \cdot b_i$ their inner product.

- $\mathbf{a} \circ \mathbf{b} = (a_i b_i)_{i\in[n]}$ the vector in $\mathbb{F}^n$ of their pairwise product, also called the Hadamard product of the two vectors.

For $\mathbf{A} \in \mathbb{F}^{n_1 n_2}, \mathbf{B} \in \mathbb{F}^{n_3 n_4}$, we denote $\mathbf{A} \otimes \mathbf{B} \in \mathbb{F}^{n_1 n_3 \times n_2 n_4}$ the Kronecker product of the two matrices. That is, if $\mathbf{A} = (a_{i,j})_{i \in [n_1], j \in [n_2]}$ then

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{1,1}\mathbf{B} & \cdots & a_{1,n_2}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{n_1,1}\mathbf{B} & \cdots & a_{n_1,n_2}\mathbf{B} \end{pmatrix}$$

We recall the mixed product of the Kronecker product stating that

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD})$$

whenever $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ have appropriate dimensions.

Sub-vectors and Sub-matrices. Let $S = \{s_1, \ldots, s_t\} \subseteq [n]$, $\overline{S} = \{\overline{s}_1, \ldots, \overline{s}_{n-t}\}$ the set $[n] \setminus S$[1]. When $\mathbf{x} = (x_1, \ldots, x_n)$ (resp. $\mathbf{G} = \begin{pmatrix} \mathbf{g}_1 & \cdots & \mathbf{g}_n \end{pmatrix}$), we denote $\mathbf{x}_S = (x_s)_{s \in S}$ (resp. $\mathbf{G}_S = \begin{pmatrix} \mathbf{g}_s \end{pmatrix}_{s \in S}$). We also use an algebraic notation for the sub-vector $\mathbf{x}_S = (x_s)$ and sub-matrix $\mathbf{G}_S$. Let $\mathbf{P}_S \in \{0,1\}^{n \times n}$ be the permutation matrix defining the ordering $s_1, \ldots, s_t, \overline{s}_1, \ldots, \overline{s}_{n-t}$. That is, $\mathbf{P}_S \mathbf{e}_{s_i} = \mathbf{e}_i$ and $\mathbf{P}_{\overline{S}} \mathbf{e}_{\overline{s}_i} = \mathbf{e}_{i+t}$, where $\mathbf{e}_i$ is the i-th unitary vector of size n. We may simply write $\mathbf{P}$ when n, S are clear from the context. For fixed $n, t \in \mathbb{N}$, we also define the matrix $\mathbf{\Sigma}_S = \begin{pmatrix} \mathbf{I}_t & \mathbf{0}_{t \times n-t} \end{pmatrix}$. Again, we may omit the subscript when the values are clear from the context. The reason to use these matrices is because $\mathbf{x}_S = \mathbf{\Sigma}_S \mathbf{P}_S \mathbf{x}$ (similarly for matrices). Finally, for $S' \subseteq S$, we write $\mathbf{x}_{S|S'}$ to denote $(\mathbf{x}_S)'_S$, that is, the restriction of $\mathbf{x}_S$ to S'. Note that $\mathbf{x}_{S'} = \mathbf{x}_{S|S'}$. We define similarly $\mathbf{G}_{S|S'}$ and $\mathbf{P}_{S|S'}$, $\mathbf{\Sigma}_{S|S'}$.

We next state a fact about subvectors and submatrices.

Fact 1. For any $\mathbf{x} \in \mathbb{F}^n$ and any $S' \subseteq S \subseteq [n]$ it holds that:

1. $\mathbf{P}_S \mathbf{x} = \begin{pmatrix} \mathbf{x}_S \\ \mathbf{x}_{\overline{S}} \end{pmatrix}$ and $\mathbf{G}\mathbf{P}_S^\top = \begin{pmatrix} \mathbf{G}_S & \mathbf{G}_{\overline{S}} \end{pmatrix}$
2. $\mathbf{x}_S = \mathbf{\Sigma}_S \mathbf{P}_S \mathbf{x}$ and $\mathbf{G}_S = \mathbf{G}\mathbf{P}_S^\top \mathbf{\Sigma}_S^\top$.
3. $\mathbf{G}\mathbf{x} = \mathbf{G}_S \mathbf{x}_S + \mathbf{G}_{\overline{S}} \mathbf{x}_{\overline{S}}$.
4. $\mathbf{x}_{S|S'} = \mathbf{x}_{S'}$ and $\mathbf{G}_{S|S'} = \mathbf{G}_{S'}$.

We extend this notation for the case where we index $n_1 n_2$-dimensional vectors using the set pair $[n_1], [n_2]$. Let $S_1 \subseteq [n_1], S_2 \subseteq [n_2]$. For $\mathbf{x} \in \mathbb{F}^{n_1 n_2}$ define $\mathbf{x}_{S_1,S_2} \in \mathbb{F}^{|S_1| \cdot |S_2|}$ as $\mathbf{x}_{S_1,S_2} = (\mathbf{x}_{(i-1)n_2+j})_{(i,j) \in (S_1 \times S_2)}$. For matrices we define $\mathbf{G}_{S_1,S_2} = (\mathbf{g}_{\ell,(i-1)n_2+j})_{(i,j) \in (S_1 \times S_2)} \in \mathbb{F}^{k \times |S_1| \cdot |S_2|}$, where k is the number of rows of $\mathbf{G}$. Similarly as before, the following holds.

Fact 2. For any $\mathbf{x} \in \mathbb{F}^{n_1 n_2}$ and any $S'_1 \subseteq S_1 \subseteq [n_1]$, $S'_2 \subseteq S_2 \subseteq [n_2]$ it holds that:

[1]We identify vector coordinates with elements of the set $[n]$. However, we can use any n-sized set with a fixed ordering. The notation we introduced extends naturally in this case.

1. For some permutation matrix $\mathbf{\Pi} \in \mathbb{F}^{n_1 n_2 \times n_1 n_2}$,

$$(\mathbf{P}_{S_1} \otimes \mathbf{P}_{S_2})\mathbf{x} = \mathbf{\Pi} \begin{pmatrix} \mathbf{x}_{S_1,S_2} \\ \mathbf{x}_{S_1,\overline{S}_2} \\ \mathbf{x}_{\overline{S}_1,S_2} \\ \mathbf{x}_{\overline{S}_1,\overline{S}_2} \end{pmatrix}, \qquad \mathbf{G}(\mathbf{P}_{S_1}^\top \otimes \mathbf{P}_{S_2}^\top) = \begin{pmatrix} \mathbf{G}_{S_1,S_2} & \mathbf{G}_{S_1,\overline{S}_2} & \mathbf{G}_{\overline{S}_1,S_2} & \mathbf{G}_{\overline{S}_1,\overline{S}_2} \end{pmatrix} \mathbf{\Pi}^\top$$

2. $\mathbf{x}_{S_1,S_2} = (\mathbf{\Sigma}_{S_1} \otimes \mathbf{\Sigma}_{S_2})(\mathbf{P}_{S_1} \otimes \mathbf{P}_{S_2})\mathbf{x}$ and $\mathbf{G}_{S_1,S_2} = \mathbf{G}(\mathbf{P}_{S_1}^\top \otimes \mathbf{P}_{S_2}^\top)(\mathbf{\Sigma}_{S_1}^\top \otimes \mathbf{\Sigma}_{S_2}^\top)$.
3. $\mathbf{G}\mathbf{x} = \mathbf{G}_{S_1,S_2}\mathbf{x}_{S_1,S_2} + \mathbf{G}_{S_1,\overline{S}_2}\mathbf{x}_{S_1,\overline{S}_2} + \mathbf{G}_{\overline{S}_1,S_2}\mathbf{x}_{\overline{S}_1,S_2} + \mathbf{G}_{\overline{S}_1,\overline{S}_2}\mathbf{x}_{\overline{S}_1,\overline{S}_2}$.
4. $\mathbf{x}_{S_1,S_2|S_1',S_2'} = \mathbf{x}_{S_1',S_2'}$ and $\mathbf{G}_{S_1,S_2|S_1',S_2'} = \mathbf{G}_{S_1',S_2'}$

Finally, we state a simple fact stating that we can derive $\mathbf{M} \otimes \mathbf{N}$ from $\mathbf{N} \otimes \mathbf{M}$ by multiplying on the left and right with appropriate permutation matrices.

Fact 3. For every $m_1, m_2, n_1, n_2 \in \mathbb{N}$ there exist permutation matrices $\mathbf{\Pi}_1 \in \{0,1\}^{m_1 m_2 \times m_1 m_2}$, $\mathbf{\Pi}_2 \in \{0,1\}^{n_1 n_2 \times n_1 n_2}$ such that for any pair of matrices $\mathbf{M} \in \mathbb{F}^{m_1 \times n_1}, \mathbf{N} \in \mathbb{F}^{m_2 \times n_2}$ it holds that $\mathbf{M} \otimes \mathbf{N} = \mathbf{\Pi}_1(\mathbf{N} \otimes \mathbf{M})\mathbf{\Pi}_2$. Matrices $\mathbf{\Pi}_1$ and $\mathbf{\Pi}_2$ depend only on the size of $\mathbf{M}$ and $\mathbf{N}$.

In cases where we iteratively halve a vector $\mathbf{x}$, we denote with $\mathbf{x_b}$ the subvector of $\mathbf{x}$ with indices prefixed with $\mathbf{b}$. With this notation, if $\mathbf{x} \in \mathbb{F}^n$, then $\mathbf{x}_0$ corresponds to the first $n/2$ elements of $\mathbf{x}$, $\mathbf{x}_1$ to the second half, $\mathbf{x}00$ to the first $n/4$ and so on.

Groups. We use implicit group notation. Let $\mathsf{gk} = (p, \mathbb{G}, \mathcal{P}) \leftarrow \mathcal{G}(1^\kappa)$ be the description of a group of size $p = \mathcal{O}(2^\kappa)$ with generator $\mathcal{P}$. Let $\mathbb{F} = \mathbb{Z}_p$. We denote $[r] = r\mathcal{P}$. We extend this notation for matrices and vectors. For a vector $a = (a_1, \ldots, a_n)$ and matrix $\mathbf{A}$ with columns $\mathbf{a}_1, \ldots, \mathbf{a}_k$, we denote

$$[\mathbf{a}] = ([a_1], \ldots, [a_n]), \qquad [\mathbf{A}] = \begin{pmatrix} [\mathbf{a}_1] & \cdots & [\mathbf{a}_k] \end{pmatrix}$$

We also consider bilinear groups, that is groups equipped with a bilinear map and extend the above notation to these. Let $\mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathcal{P}_1, \mathcal{P}_2) \leftarrow \mathcal{G}(1^\kappa)$ be the description of an asymmetric bilinear group of size $p = \mathcal{O}(2^\kappa)$ equipped with an efficiently computable bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $\mathcal{P}_\mu$ is a generator of $\mathbb{G}_\mu$ for $\mu \in \{1,2\}$. For $r \in \mathbb{F}$ we denote $[r]_\mu = r\mathcal{P}_\mu$ for $\mu \in \{1,2,T\}$ and $\mathcal{P}_T = e(\mathcal{P}_1, \mathcal{P}_2)$. In this notation we $e([r]_1, [s]_2) = [rs]_T$.

2.2 Polynomials and the Lagrange Basis

Lagrange basis. Let $\mathbb{H} = \{\mathsf{h}_1, \ldots, \mathsf{h}_m\}$ be a multiplicative group of size m in $\mathbb{F}$[2]. We consider the set of Lagrange interpolation polynomials $\left\{\lambda_j^{\mathbb{H}}(X)\right\}_{j=1}^m$ associated with $\mathbb{H}$,

[2]Working in multiplicative subgroups of $\mathbb{F}$ is not necessary, but is often needed for improved efficiency

namely,

$$\lambda_j^{\mathbb{H}}(X) = \prod_{i \neq j} \frac{X - \mathsf{h}_i}{\mathsf{h}_j - \mathsf{h}_i}$$

Recall that $\sum_{j=1}^{m} \lambda_j^{\mathbb{H}}(X) = 1$. Moreover, we define $t(X) = \prod_{j=1}^{m}(X - \mathsf{h}_j)$ the vanishing polynomial in $\mathbb{H}$.

For the multivariate case, recall that $\lambda_{\boldsymbol{\sigma}}^{\mathbb{H}}(X_\nu, \ldots, X_1) = \prod_{j=1}^{\nu} \lambda_j(X_j)$ where $\boldsymbol{\sigma} \in \{[m]^n\}$. Using this, we can write the vector of multivariate Lagrange polynomials as the Kronecker product $\boldsymbol{\lambda}^{\mathbb{H}}(X_\nu) \otimes \cdots \otimes \boldsymbol{\lambda}^{\mathbb{H}}(X_1)$, where $\boldsymbol{\lambda}^{\mathbb{H}}(X)$ is the univariate Lagrange basis. When $|\mathbb{H}| = 2$, we refer to the Lagrange polynomials as the *multilinear* Lagrange basis, while for $|\mathbb{H}| > 2$, we refer to them as the *low degree* Lagrange basis.

2.3 Cryptographic Assumptions

We next introduce the cryptographic assumptions we will use throughout this thesis.

Assumption 1. Let $\mathcal{U}_{n,m,r}$ be the distribution that outputs uniform rank r matrices of dimension $n \times m$ over $\mathbb{F}$. The (n, m)-Rank Assumption [Vil12] holds in $\mathbb{G}$ if for all $1 \leq r_1 < r_2 \leq \min(n, m)$ and for all non-uniform PPT adversaries $\mathcal{A}$ and relative to $\mathsf{gk} \leftarrow \mathcal{G}(1^\kappa)$ and the coin tosses of adversary $\mathcal{A}$,

$$\Pr\left[\mathcal{A}(\mathsf{gk}, [\mathbf{U}_1]) = 1 \mid \mathbf{U}_1 \leftarrow \mathcal{U}_{n,m,r_1}\right] - \Pr\left[\mathcal{A}(\mathsf{gk}, [\mathbf{U}_2]) = 1 \mid \mathbf{U}_1 \leftarrow \mathcal{U}_{n,m,r_2}\right] \leq \mathsf{negl}(\kappa)$$

As shown in [Vil12], the Rank assumption reduces to DDH (we define it next in terms of matrix distributions).

We next recall two q-type assumption, the generalized Discrete Logarithm assumption ((q_1, q_2)-DLOG) and the Bilinear Strong Diffie Hellman assumption ((q_1, q_2)-BSDH) [BB11]. When $q_1 = q_2$, we simply call them q-DLOG and q-BSDH respectively. We present them next.

Definition 1. The (q_1, q_2)-DLOG assumption holds relative to $\mathcal{G}(1^\lambda)$ if for all PPT adversaries $\mathcal{A}$, the following probability is negligible in λ.

$$\Pr\left[\tau \leftarrow\!\!\leftarrow \mathcal{A}(\mathsf{gk}, \left\{[\tau^i]_1\right\}_{i=0}^{q_1}, \left\{[\tau^i]_2\right\}_{i=0}^{q_2}) \,\middle|\, \mathsf{gk} \leftarrow \mathcal{G}(1^\kappa);\ \tau \leftarrow \mathbb{F}\right].$$

Definition 2. The (q_1, q_2)-BSDH assumption holds relative to $\mathcal{G}(1^\lambda)$ if for all PPT adversaries $\mathcal{A}$, the following probability is negligible in λ.

$$\Pr\left[(c, \tfrac{1}{(\tau - c)} e\left([1]_1, [1]_2\right) \leftarrow \mathcal{A}(\mathsf{gk}, \left\{[\tau^i]_1\right\}_{i=0}^{q_1}, \left\{[\tau^i]_2\right\}_{i=0}^{q_2}) \,\middle|\, \mathsf{gk} \leftarrow \mathcal{G}(1^\kappa);\ \tau \leftarrow \mathbb{F}\right].$$

We next recall the definition of a matrix distribution [EHK+13] and present related assumptions.

Definition 3. Let $k, \ell \in \mathbb{N}$. We call $\mathcal{D}_{\ell,k}$ (resp. $\mathcal{D}_k$) a matrix distribution if it outputs in PPT time with overwhelming probability matrices in $\mathbb{F}^{\ell \times k}$ (resp. in $\mathbb{F}^{(k+1)\times k}$). For a matrix distribution $\mathcal{D}_k$, we denote as $\overline{\mathcal{D}}_k$ the distribution of the first k rows of the matrices sampled according to $\mathcal{D}_k$.

We next present assumptions about matrix distributions.

Assumption 2. Let $\mathcal{D}_{\ell,k}$ be a matrix distribution and $\gamma \in \{1, 2\}$. For all non-uniform PPT adversaries $\mathcal{A}$ and relative to $\mathsf{gk} \leftarrow \mathbb{G}(1^\kappa)$, $\mathbf{A} \leftarrow \mathcal{D}_{\ell,k}$ and the coin tosses of adversary $\mathcal{A}$, the Find-Rep Assumption [EHK+13] in $\mathbb{G}_\gamma$ ($\mathcal{D}_{\ell,k}$-$\mathsf{FindRep}_\gamma$) holds if

$$\Pr\left[\ \mathbf{r} \leftarrow \mathcal{A}(\mathsf{gk}, [\mathbf{A}]_\gamma) = 1 : \mathbf{r} \neq \mathbf{0} \wedge \mathbf{r}^\top \mathbf{A} = \mathbf{0}\ \right] \leq \mathsf{negl}(\kappa)$$

Assumption 3. Let $\mathcal{D}_{\ell,k}$ be a matrix distribution. For all non-uniform PPT adversaries $\mathcal{A}$ and relative to $\mathsf{gk} \leftarrow \mathcal{G}(1^\kappa)$, $\mathbf{A} \leftarrow \mathcal{D}_{\ell,k}$ and the coin tosses of adversary $\mathcal{A}$,

1. the Kernel Matrix Diffie-Hellman Assumption [MRV16] ($\mathcal{D}_{\ell,k}$-$\mathsf{SKerMDH}_\gamma$) holds in $\mathbb{G}_\gamma$ if
$$\Pr\left[[\mathbf{r}]_{3-\gamma} \leftarrow \mathcal{A}(\mathsf{gk}, [\mathbf{A}]_\gamma) : \mathbf{r} \neq \mathbf{0} \wedge \mathbf{r}^\top \mathbf{A} = 0\right] \leq \mathsf{negl}(\kappa)$$
2. the Split Kernel Matrix Diffie-Hellman Assumption [GHR15] ($\mathcal{D}_{\ell,k}$-$\mathsf{SKerMDH}$) holds if
$$\Pr\left[[\mathbf{r}]_1, [\mathbf{s}]_2 \leftarrow \mathcal{A}(\mathsf{gk}, [\mathbf{A}]_1, [\mathbf{A}]_2) : \mathbf{r} \neq \mathbf{s} \wedge \mathbf{r}^\top \mathbf{A} = \mathbf{s}^\top \mathbf{A}\right] \leq \mathsf{negl}(\kappa)$$

Assumption 4. Let $\mathcal{D}_{\ell,k}$ be a matrix distribution and $\mathsf{gk} \leftarrow \mathbb{G}(1^\kappa)$. For all non-uniform PPT adversaries $\mathcal{A}$ and relative to $\mathsf{gk} \leftarrow \mathbb{G}(1^\kappa)$, $\mathbf{A} \leftarrow \mathcal{D}_{\ell,k}, \mathbf{w} \leftarrow \mathbb{F}^k, [\mathbf{z}]_\gamma \leftarrow \mathbb{G}_\gamma^\ell$ and the coin tosses of adversary $\mathcal{A}$, the Matrix Decisional Diffie-Hellman Assumption [EHK+13] in $\mathbb{G}_\gamma$ ($\mathcal{D}_{\ell,k}$-MDDH_γ) holds if

$$\left|\Pr[\mathcal{A}(\mathsf{gk}, [\mathbf{A}]_\gamma, [\mathbf{Aw}]_\gamma) = 1] - \Pr[\mathcal{A}(gk, [\mathbf{A}]_\gamma, [\mathbf{z}]_\gamma) = 1]\right| \leq \mathsf{negl}(\kappa)$$

A lot of assumptions can be abstracted in the Matrix assumption framework. We next consider some distributions and discuss the hardness assumption associated with them:

- We denote with $\mathcal{U}_{\ell,k}$ the distribution that outputs uniformly distributed matrices over $\mathbb{F}^{\ell \times k}$. The discrete logarithm, computational and decisional diffie hellman assumptions (or more accurately slighly stronger variation of these) in $\mathbb{G}_\mu$ can be restated as the $\mathcal{U}_{1\times 2}$-FindRep, $\mathcal{U}_{1\times 3}$-MDDH assumptions respecitvely.
- We denote with $\mathcal{L}_k$ the distribution that outputs $k + 1 \times k$ matrices where the first k rows are $\mathbf{e}_1, \ldots, \mathbf{e}_k$ and the last row is uniformly distributed. The $\mathcal{L}_2$-MDDH is known as the DLin assumptionknown as the DLin assumption.
- Let $\mathbf{m}(X) = (1, X, \ldots, X^{n-1}) \in \mathbb{F}[X]^n$. We denote with $\mathcal{X}_{\ell,m}$ the matrix distribution that samples $\tau_1, \ldots, \tau_\ell \leftarrow \mathbb{F}$ and outputs the $\ell \times n$ matrix with rows $\mathbf{m}(s_i)$. The $\mathcal{X}_{\ell,m}$-MDDH assumption reduces to the q-DLog assumption. The corresponding kernel and matrix assumption do not hold for $\ell < 2$. For $\ell \geq 2$ they hold generically [GR19].

- Similarly, let $\boldsymbol{\lambda}(X) = (\lambda_1(X), \ldots, \lambda_m(X)) \in \mathbb{F}[X]^m$ be the vector of lagrange polynomials over as set $\mathbb{H}$ of size m. We denote with $\mathcal{LG}^{\mathbb{H}}_{\ell,m}$ the matrix distribution that samples $\tau_1, \ldots, \tau_\ell \leftarrow \mathbb{F}$ and outputs the $\ell \times n$ matrix with rows $\boldsymbol{\lambda}(s_i)$. Note that this distribution is linearly related to $\mathcal{X}_{\ell,m}$, so they are equivalent from a cryptographic point of view.

- We extend the two previous assumptions to the multivariate case. We denote them $\mathcal{X}_{\nu,\ell,m}$, $\mathcal{LG}^{\mathbb{H}}_{\nu,\ell,m}$ respectively, where the first index denotes the nubmer of variables, the second the secret points on which we evaluate and the third the individual degree.

We recall here that the $\mathcal{X}_{\nu,1,1}$ assumption (the evaluation of ν-variate Lagrange polynomial in a secret point) reduces to the 1-DLOG assumption.

We also introduce a new assumption called the *Kronecker* MDDH assumption.

Assumption 5. Let $\mathcal{U}_{\ell,k}, \mathcal{V}_{\ell,k}$ be matrix distributions and $\mathsf{gk} \leftarrow \mathbb{G}(1^\kappa)$. The $(\mathcal{U} \otimes \mathcal{V})$-MDDH assumption holds if

1. For all non-uniform PPT adversaries $\mathcal{A}$ and relative to $\mathsf{gk} \leftarrow \mathbb{G}(1^\kappa)$, $\mathbf{U} \leftarrow \mathcal{U}_{\ell,k}, \mathbf{V} \leftarrow \mathcal{V}_{\ell,k}, \mathbf{R} \leftarrow \mathbb{F}^{\ell^2 \times k^2}, \mathbf{k} \leftarrow \mathbb{F}^{k^2}, \mathbf{r} \leftarrow \mathbb{F}^{\ell^2}, \mathbf{s}, \mathbf{t} \in\leftarrow \mathbb{F}^{\ell^2}$ and the coin tosses of adversary $\mathcal{A}$,

$$\Big|\Pr[\mathcal{A}(\mathsf{gk}, [\mathbf{U}]_1, [\mathbf{V}]_2, [\mathbf{U}\otimes\mathbf{V} - \mathbf{R}]_1, [\mathbf{R}]_2, [(\mathbf{U}\otimes\mathbf{V})\mathbf{k} - \mathbf{r}]_1, [\mathbf{r}]_2) = 1] - \Pr[\mathcal{A}(\mathsf{gk}, [\mathbf{U}]_1, [\mathbf{V}]_2, [\mathbf{U}\otimes\mathbf{V} - \mathbf{R}]_1, [\mathbf{R}]_2, [\mathbf{s}]_1, [\mathbf{t}]_2) = 1]\Big| \leq \mathsf{negl}(\kappa)$$

2. For all non-uniform PPT adversaries $\mathcal{A}$ and relative to $\mathsf{gk} \leftarrow \mathbb{G}(1^\kappa)$, $\mathbf{U} \leftarrow \mathcal{U}_{\ell,k}, \mathbf{V} \leftarrow \mathcal{V}_{\ell,k}, \mathbf{R} \leftarrow \mathbb{F}^{\ell^2 \times k^2}, \mathbf{k} \leftarrow \mathbb{F}^{\ell}, \mathbf{l} \leftarrow \mathbb{F}^{\ell}, \mathbf{s}, \mathbf{t} \in\leftarrow \mathbb{F}^{k}$ and the coin tosses of adversary $\mathcal{A}$,

$$\Big|\Pr[\mathcal{A}(\mathsf{gk}, [\mathbf{U}]_1, [\mathbf{V}]_2, [\mathbf{U}\otimes\mathbf{V} - \mathbf{R}]_1, [\mathbf{R}]_2, [\mathbf{U}\mathbf{k}]_1, [\mathbf{V}\mathbf{l}]_2) = 1] - \Pr[\mathcal{A}(\mathsf{gk}, [\mathbf{U}]_1, [\mathbf{V}]_2, [\mathbf{U}\otimes\mathbf{V} - \mathbf{R}]_1, [\mathbf{R}]_2, [\mathbf{s}]_1, [\mathbf{t}]_2) = 1]\Big| \leq \mathsf{negl}(\kappa)$$

The former condition states that a random vector in the image $\mathbf{U} \otimes \mathbf{V}$ split in $\mathbb{G}_1, \mathbb{G}_2$ is pseudorandom, even if the adversaries knows $[\mathbf{U}]_1, [\mathbf{V}]_2$. The latter that $\mathcal{U}$-MDDH$_1$ and $\mathcal{V}$-MDDH$_2$ hold even when we give to the adversary the split of $\mathbf{U} \otimes \mathbf{V}$ in the two groups. We next show that the assumption reduces to $\mathcal{U}$-MDDH$_1$, $\mathcal{V}$-MDDH$_2$ and DDH in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

Theorem 1. *The* $(\mathcal{U} \otimes \mathcal{V})$-MDDH *assumption holds if* $\mathcal{U}$-MDDH$_1$ *and* $\mathcal{V}$-MDDH$_2$ *assumptions hold, and* DDH *assumption holds in both* $\mathbb{G}_1$ *and* $\mathbb{G}_2$.

Proof. The second conditions is immediate by a hybrid argument. It is enough to note that we can compute the splitting of $\mathbf{U} \otimes \mathbf{V}$ in the two groups if we know the discrete log of either $\mathbf{U}$ or $\mathbf{V}$, that is, we can compute $[\mathbf{U}]_1 \otimes \mathbf{V}$ and $\mathbf{U} \otimes [\mathbf{V}]_2$.

For the first part, we need to show that the distributions

$$[\mathbf{U}]_1, [\mathbf{V}]_2, [\mathbf{U} \otimes \mathbf{V} - \mathbf{R}]_1, [\mathbf{R}]_2, [(\mathbf{U} \otimes \mathbf{V})^\top \mathbf{k} - \mathbf{r}]_1, [\mathbf{r}]_2 : \mathbf{k} \leftarrow \mathbb{F}^{k^2}; \mathbf{r} \leftarrow \mathbb{F}^{\ell^2}$$

$$[\mathbf{U}]_1, [\mathbf{V}]_2, [\mathbf{U} \otimes \mathbf{V} - \mathbf{R}]_1, [\mathbf{R}]_2, [\mathbf{s}]_1, [\mathbf{t}]_2 : \mathbf{s}, \mathbf{t} \leftarrow \mathbb{F}^{\ell^2}$$

are computationally indistuinguishable.

We show the indistinguishability of these distributions by showing indistinguishability of a sequence of hybrid distributions. In what follows denote $\alpha = ([\mathbf{U}]_1, [\mathbf{V}]_2, [\mathbf{U} \otimes \mathbf{V} - \mathbf{R}]_1, [\mathbf{R}]_2)$.

We have

0. $\alpha, [(\mathbf{U} \otimes \mathbf{V})\mathbf{k} - \mathbf{r}]_1, [\mathbf{r}]_2 :$ $\mathbf{r} \leftarrow \mathbb{F}^{\ell^2}, \mathbf{k} \leftarrow \mathbb{F}^{k^2}$
1. $\alpha, [(\mathbf{U} \otimes \mathbf{V})(\mathbf{k}_1 \otimes \mathbf{k}_2) - \mathbf{r}]_1, [\mathbf{r}]_2 :$ $\mathbf{r} \leftarrow \mathbb{F}^{\ell^2}, \quad \mathbf{k}_1, \mathbf{k}_2 \leftarrow \mathbb{F}^{k}$
2. $\alpha, [(\mathbf{U}\mathbf{k}_1) \otimes (\mathbf{V}\mathbf{k}_2) - \mathbf{r}]_1, [\mathbf{r}]_2 :$ $\mathbf{r} \leftarrow \mathbb{F}^{\ell^2}, \quad \mathbf{k}_1, \mathbf{k}_2 \leftarrow \mathbb{F}^{k}$
3. $\alpha, [\mathbf{u} \otimes (\mathbf{V}\mathbf{k}_2) - \mathbf{r}]_1, [\mathbf{r}]_2 :$ $\mathbf{u} \leftarrow \mathbb{F}^{\ell}, \quad \mathbf{r} \leftarrow \mathbb{F}^{\ell^2}, \quad \mathbf{k}_2 \leftarrow \mathbb{F}^{k}$
4. $\alpha, [\mathbf{r}]_1, [\mathbf{u} \otimes (\mathbf{V}\mathbf{k}_2) - \mathbf{r}]_2 :$ $\mathbf{u} \leftarrow \mathbb{F}^{\ell}, \quad \mathbf{r} \leftarrow \mathbb{F}^{\ell^2}, \quad \mathbf{k}_2 \leftarrow \mathbb{F}^{k}$
5. $\alpha, [\mathbf{r}]_1, [\mathbf{u} \otimes \mathbf{v} - \mathbf{r}]_2 :$ $\mathbf{u}, \mathbf{v} \leftarrow \mathbb{F}^{\ell}, \quad \mathbf{r} \leftarrow \mathbb{F}^{\ell^2}$
6. $\alpha, [\mathbf{s}]_1, [\mathbf{t}]_2 :$ $\mathbf{s}, \mathbf{t} \leftarrow \mathbb{F}^{\ell}$

We next show that for all $1 \leq i \leq 5$ the distributions $i-1, i$ are computationally indistinguishable.

- *Case* $i = 1$. We show that distinguishing these two distributions reduces to the (n, n)-Rank assumption in $\mathbb{G}_1$. Assume there exists a distinguisher $\mathcal{A}$ for distributions 0 and 1. We construct a distinguisher $\mathcal{B}$ against the Rank assumption. The distinguisher works as follows: on input $[\mathbf{A}]_1 \in \mathbb{F}^{k \times k}$, it samples $\mathbf{U} \leftarrow \mathcal{U}, \mathbf{V} \leftarrow \mathcal{V}, \mathbf{R} \leftarrow \mathbb{F}^{\ell^2 \times k^2}, \mathbf{r} \leftarrow \mathbb{F}^{\ell^2}$. It computes $[\mathbf{M}]_1 = \mathbf{U}[\mathbf{A}]_1\mathbf{V}^\top$ and vectorizes it; denote the vectorization as $[\mathbf{m}]_1$. it then executes

  $$\mathcal{A}([\mathbf{U}]_1, [\mathbf{V}]_2, [\mathbf{U} \otimes \mathbf{V} - \mathbf{R}]_1, [\mathbf{R}]_2, [\mathbf{m}]_1 - [\mathbf{r}]_1, [\mathbf{r}]_2)$$

  and outputs whatever $\mathcal{A}$ outputs. Now, note the vectorization $[\mathbf{m}]_1$ corresponds to the value $[(\mathbf{U} \otimes \mathbf{V})\mathbf{m}]_1$. If $[\mathbf{A}]$ is of rank 1, then we can write $\mathbf{A} = \mathbf{k}_1\mathbf{k}_2^\top$ and we have $\mathbf{M} = \mathbf{U}\mathbf{k}_1\mathbf{k}_2\mathbf{V}^\top = (\mathbf{U}\mathbf{k}_1) \otimes (\mathbf{V}\mathbf{k}_2)$ and the vectorization corresponds to $(\mathbf{U}\mathbf{k}_1) \otimes (\mathbf{V}\mathbf{k}_2)$, namely the case $i = 0$. Otherwise, $[\mathbf{A}]$ is of rank n, and we can write its vectorization as $\mathbf{k}$. Then, $\mathbf{m}$ correspond to $(\mathbf{U} \otimes \mathbf{V})\mathbf{k}$, namely the case $i = 1$.

- *Case* $i = 2$. Distributions $1, 2$ are perfectly indistinguishability since the only difference is that the latter is computed as $[(\mathbf{U}\mathbf{k}_1) \otimes (\mathbf{V}\mathbf{k}_2) - \mathbf{r}]_1$, which equals to

$$[(\mathbf{U} \otimes \mathbf{V})(\mathbf{k}_1 \otimes \mathbf{k}_2) - \mathbf{r}]_1$$

  which is the corresponding value of distribution 1.

- *Case* $i = 3$. This case reduces to the $\mathcal{U}$-MDDH$_1$ assumption. The only difference is that in the forth distribution, we replace $\mathbf{U}\mathbf{k}_1$ with a uniform element $\mathbf{u}$. It is enough to show that we can compute the rest of the values given $[\mathbf{U}]_1, [\mathbf{u}]_1$ where $[\mathbf{u}]$ is either $\mathbf{U}\mathbf{k}_1$ or uniform. We can compute the values as

$$[\mathbf{U}]_1, [\mathbf{V}]_2, [\mathbf{U}]_1 \otimes \mathbf{V} - [\mathbf{R}]_1, [\mathbf{R}]_2, [\mathbf{u}]_1 \otimes (\mathbf{V}\mathbf{k}_2) - [\mathbf{r}]_1, [\mathbf{r}]_2$$

  where we sample $\mathbf{V} \leftarrow \mathcal{V}_S, \mathbf{R} \leftarrow \mathbb{F}^{\ell^2 \times k^2}, \mathbf{r} \leftarrow \mathbb{F}^{\ell^2}, \mathbf{k}_2 \leftarrow \mathbb{F}^{\ell}$.

- *Case* $i = 4$. The distributions 4 and 5 are perfectly indistinguishable. It is enough to note that in both, the last two elements are uniformly distributed conditioned on their sum of discrete logarithms being equal to $\mathbf{u} \otimes (\mathbf{V}\mathbf{k}_2)$.

- *Case* $i = 5$. This is the same as the case $i = 3$ for the value $[\mathbf{v}]_2$. This case reduces to the $\mathcal{V}$-MDDH$_2$ assumption. The only difference is that in the last distribution, we replace $\mathbf{V}\mathbf{k}_2$ with a uniform element $\mathbf{v}$. It is enough to show that we can compute the rest of the values given $[\mathbf{V}]_2, [\mathbf{v}]_2$ where $\mathbf{v}$ is either $\mathbf{V}\mathbf{k}_2$ or uniform. We can compute the values as

$$[\mathbf{U}]_1, [\mathbf{V}]_2, [\mathbf{R}]_1, \mathbf{U} \otimes [\mathbf{V}]_2 - [\mathbf{R}]_2, [\mathbf{r}]_1, \mathbf{u} \otimes [\mathbf{v}]_2 - [\mathbf{r}]_2$$

  where we sample $\mathbf{U} \leftarrow \mathcal{U}_S, \mathbf{R} \leftarrow \mathbb{F}^{\ell^2 \times k^2}, \mathbf{r} \leftarrow \mathbb{F}^{\ell^2}, \mathbf{u} \leftarrow \mathbb{F}^{\ell}$.

- *Case* $i = 6$. This again reduces to the Rank assumption in $\mathbb{G}_2$. The only difference in the two distributions is that in distribution 5 the sum of the last two elements, namely $\mathbf{u} \otimes \mathbf{v}$ is a vectorized matrix of rank 1, namely $\mathbf{u}\mathbf{v}$, while in distribution 6 is a uniformly distributed matrix of rank n (except w.n.p). Given $[\mathbf{A}]_2 \in \mathbb{G}_2^{n \times n}$ either uniform of rank 1 or uniform of rank n we can compute all the other values efficiently as follows. Let $\mathbf{a}$ be the vectorization of $\mathbf{T}$. We compute

$$[\mathbf{U}]_1, [\mathbf{V}]_2, [\mathbf{U} \otimes \mathbf{V} - \mathbf{R}]_1, [\mathbf{R}]_2, [\mathbf{r}]_1, [\mathbf{a}]_2 - [\mathbf{r}]_2$$

  where $\mathbf{U} \leftarrow \mathcal{U}_S, \mathbf{V} \leftarrow \mathcal{V}_S, \mathbf{R} \leftarrow \mathbb{F}^{\ell^2 \times k^2}, \mathbf{r} \leftarrow \mathbb{F}^{k^2}$.

□

Algebraic Group Model (AGM). The algebraic group model [FKL18] lies between the standard model and the stronger generic group model [Sho97]. In AGM, we consider only so-called algebraic adversaries. Such adversaries have direct access to group elements and, in particular, can use their bit representation, like in the standard model. However,

these adversaries are assumed to output new group elements only by applying the group operation to received group elements, similarly to the generic group model. This requirement is formalized as follows: Suppose an adversary $\mathcal{A}$ is given some group elements $[x_1]_1 \dots [x_m]_1 \in \mathbb{G}_1$. Then, for every new group element $[z]_1 \in \mathbb{G}_1$ that the adversary outputs, it must also output $z_1 \dots z_m \in \mathbb{F}$ such that $[z]_1 = \sum_{i=1}^{m} z_i [x_i]_1$.

2.4 Commitment Schemes

In this section we define commitment schemes. This is a very useful cryptographic primitive that -as the name suggests- allows a prover to first commit to a value m and later reveal it. The prover should not be able to change its mind between committing and revealing. This property is called binding. Optionally, the commitment is hiding, meaning that it does not leak information about the committed message.

Definition 4 (Commitment Scheme)**.** A commitment scheme CS is a tuple of PPT algorithms $(\mathsf{KeyGen}, \mathsf{Com}, \mathsf{Open},)$ that work as follows:

$\mathsf{ck} \leftarrow \mathsf{CS.KeyGen}(1^\kappa)$**:** On input the security parameter κ, it outputs a commitment key ck. The commitment key defines a messagespace $\mathcal{M}$ and commitment space C.

$(\mathsf{c}, \pi) \leftarrow \mathsf{CS.Com}(\mathsf{ck}, \mathsf{m}; \rho)$**:** On input ck and a message $\mathsf{m} \in \mathcal{M}$, it outputs a commitment $\mathsf{c} \in C$ and opening information aux.

$0/1 \leftarrow \mathsf{CS.Open}(\mathsf{ck}, \mathsf{c}, \mathsf{m}, \pi)$**:** On input ck, c, m and π, it outputs a bit indicating if c is a valid commitment to m or not.

that satisfies the following properties:

1. *Correctness.* For all $\kappa \in \mathbb{N}$ and $\mathsf{m} \in \mathcal{M}$:

$$\Pr\left[\mathsf{CS.Open}(\mathsf{ck}, \mathsf{c}, \mathsf{m}, \pi) = 1 \;\middle|\; \begin{array}{c} \mathsf{ck} \leftarrow \mathsf{CS.KeyGen}(1^\kappa) \\ (\mathsf{c}, \pi) \leftarrow \mathsf{CS.Com}(\mathsf{ck}, \mathsf{m}) \end{array}\right] = 1$$

2. *Binding.* A commitment scheme is *binding* if, for all PPT adversaries $\mathcal{A}$, and all $\kappa \in \mathbb{N}$:

$$\Pr\left[\begin{array}{c} \mathsf{CS.Open}(\mathsf{ck}, \mathsf{c}, \mathsf{m}, \pi) = 1 \\ \mathsf{CS.Open}(\mathsf{ck}, \mathsf{c}, \mathsf{m}', \pi') = 1 \\ \mathsf{m} \neq \mathsf{m}' \end{array} \;\middle|\; \begin{array}{c} \mathsf{ck} \leftarrow \mathsf{CS.KeyGen}(1^\kappa) \\ (\mathsf{c}, (\mathsf{m}, \pi), (\mathsf{m}', \pi')) \leftarrow \mathcal{A}(\mathsf{ck}) \end{array}\right] \leq \mathsf{negl}(\kappa)$$

A commitment can also satisfy a privacy property called "hiding". We define it next.

Definition 5 (Hiding Commitment Scheme)**.** A commitment scheme CS is statistically hiding if for all $\mathsf{m}_0, \mathsf{m}_0 \in \mathcal{M}$ and all (even computationally unbounded) adversaries $\mathcal{D}$

$$\Pr\left[\mathcal{D}(\mathsf{ck}, \mathsf{c}_b) \;\middle|\; \begin{array}{c} \mathsf{ck} \leftarrow \mathsf{CS.KeyGen}(1^\kappa) \\ b \leftarrow \{0,1\} \\ (\mathsf{c}, \pi) \leftarrow \mathsf{CS.Com}(\mathsf{ck}, \mathsf{m}_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\kappa)$$

In most cases, the proof is simply the randomness used to construct the commitment. In this case, the verifier simply reconstructs the commitment and asserts that it is the same as the claimed one. In this case, we say that the commitment has *canonical* openings and we omit the presentation of the Open algorithm.

2.5 Non-Interactive (Zero Knowledge) Arguments of Knowledge.

We next define non-interactive zero knowledge arguments. One can consider many variations for the definition, for example whether it is for a single language or for many languages, if the setup is trusted/transparent/updateable and so on. Here, we consider a simple definition for a fixed language. We first define a non-interactive argument system (NARK) and then define succinctness and zero knowledge properties for it.

Definition 6 (Non-Interactive Arguement of Knowledge)**.** Let $\mathcal{L}$ be an **NP** language and $\mathcal{R}$ the corresponding **NP** relation. A non-interactive argument of knowledge (NARK) for $\mathcal{L}$ is a tuple of PPT algorithms $(\mathsf{KeyGen}, \mathsf{Prove}, \mathsf{Verify})$ that work as follows:

$\mathsf{srs} \leftarrow \mathsf{NARK.KeyGen}(1^\kappa)$**:** On input the security parameter κ, it outputs a structured reference string srs.

$\pi \leftarrow \mathsf{NARK.Prove}(\mathsf{srs}, \mathsf{x}, \mathsf{w})$**:** On input srs and a statement witness pair $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$, it outputs a proof π.

$0/1 \leftarrow \mathsf{NARK.Verify}(\mathsf{srs}, \mathsf{x}, \pi)$**:** On input srs, a statement x and a proof π, it outputs a bit indicating acceptance or rejection of the claim.

that satisfies the following properties:

1. *Completeness.* For all $\kappa \in \mathbb{N}$, and all $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$:

$$\Pr\left[\mathsf{NARK.Verify}(\mathsf{srs}, \mathsf{x}, \pi) = 1 \;\middle|\; \begin{array}{c} \mathsf{srs} \leftarrow \mathsf{NARK.KeyGen}(1^\kappa) \\ \pi \leftarrow \mathsf{NARK.Prove}(\mathsf{srs}, \mathsf{x}, \pi) \end{array}\right] = 1$$

2. *Knowledge Soundness.* For all $\kappa \in \mathbb{N}$, and all PPT adversaries $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}$ such that:

$$\Pr\left[\begin{array}{c|c} \mathsf{NARK.Verify}(\mathsf{srs},\mathsf{x},\pi)=1 & \mathsf{srs}\leftarrow\mathsf{NARK.KeyGen}(1^\kappa) \\ \Rightarrow & (\mathsf{x},\pi)\leftarrow\mathcal{A}(\mathsf{srs}) \\ (\mathsf{x},\mathsf{w})\in\mathcal{R} & \mathsf{w}\leftarrow\mathcal{E}^{\mathcal{A}}(\mathsf{srs}) \end{array}\right] \geq 1-\mathsf{negl}(\kappa)$$

Next we define a family of NARKs that are *succinct*, namely, the proof size is sublinear in the size of the **NP** witness.

Definition 7 (Succinct Non-Interactive Arguement of Knowledge)**.** Let $\mathcal{L}$ be an **NP** language and $\mathcal{R}$ the corresponding **NP** relation. A succinct non-interactive argument of knowledge (SNARK) is a NARK for $\mathcal{L}$ that also satisfies that the size of the proof π is sublinear in the size of the **NP** witness w.

Finally, we define the zero knowledge property.

Definition 8 (Zero Knowledge NARK or SNARK)**.** Let $\mathcal{L}$ be an **NP** language and $\mathcal{R}$ the corresponding **NP** relation. A NARK (or SNARK) is zero knowledge if there exists a pair of PPT algorithms $(\mathcal{S}_1,\mathcal{S}_2)$ such that for all $(\mathsf{x},\mathsf{w})\in\mathcal{R}$ the distributions

$$\left\{\mathsf{srs},\mathsf{x},\pi \;\middle|\; \begin{array}{c} \mathsf{srs}\leftarrow\mathsf{NARK.KeyGen}(1^\kappa) \\ \pi\leftarrow\mathsf{NARK.Prove}(\mathsf{srs},\mathsf{x},\mathsf{w}) \end{array}\right\},\qquad \left\{\mathsf{srs}_{\mathcal{S}},\mathsf{x},\pi_{\mathcal{S}} \;\middle|\; \begin{array}{c} \mathsf{srs}_{\mathcal{S}}\leftarrow\mathcal{S}_1(1^\kappa;\rho) \\ \pi_{\mathcal{S}}\leftarrow\mathcal{S}_2(\mathsf{srs}_{\mathcal{S}},\mathsf{x},\rho) \end{array}\right\}$$

are statistically indistinguishable.

2.6 Interactive (Zero Knowledge) Arguments of Knowledge.

We present the definitions and the relevant results we need for (Zero Knowledge) Arguments of Knowledge (ZKAoK). We follow the presentation of [BCC+16].

Let $\mathcal{L}\in\mathbf{NP}$ be a language and $\mathcal{R}$ the corresponding relation for $\mathcal{L}$. A ZKAoK allows a prover to convince a verifier of knowledge of a witness w certifying membership of a public x in $\mathcal{L}$ that is $(\mathsf{x},\mathsf{w})\in\mathcal{R}$. The zero knowledge property guarantees that the verifier learns nothing about the witness w apart from the fact that the prover knows such a witness.

Denote with $\langle\mathcal{P}(\mathsf{x},\mathsf{w}),\mathcal{V}(\mathsf{x})\rangle$ the transcript of an execution of $\mathcal{P}$ and $\mathcal{V}$ with respective inputs x, w and x. Let $\mathsf{view}_{\mathcal{V}}\langle\mathcal{P}(\mathsf{x},\mathsf{w}),\mathcal{V}(\mathsf{x})\rangle$ ($\mathsf{view}_{\mathcal{P}}\langle\mathcal{P}(\mathsf{x},\mathsf{w}),\mathcal{V}(\mathsf{x})\rangle$) be the views of $\mathcal{V}$ ($\mathcal{P}$) in a protocol execution (i.e. the input, randomness and all incoming messages), and finally let $\mathsf{out}_{\mathcal{V}}\langle\mathcal{P}(\mathsf{x},\mathsf{w}),\mathcal{V}(\mathsf{x})\rangle$ be the final verdict of the verifier (accept or reject).

Definition 9. The pair $\langle\mathcal{P},\mathcal{V}\rangle$ is a Zero Knowledge Argument of Knowledge if it is public coin, it has perfect completeness, statistical witness extended emulation and perfect honest verifier zero knowledge as defined next.

Definition 10. The pair $\langle \mathcal{P}, \mathcal{V} \rangle$ has Perfect Completeness if for all $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$ it holds that $\Pr\left[\mathsf{out}_{\mathcal{V}}\langle \mathcal{P}(\mathsf{x}, \mathsf{w}), \mathcal{V}(\mathsf{x})\rangle = 1\right] = 1$.

Definition 11. The pair $\langle \mathcal{P}, \mathcal{V} \rangle$ has Statistical Witness Extended Emulation if for all deterministic polynomial $\mathcal{P}^*$, there exists an expected polynomial time extractor $\mathcal{E}$, such that for all (unbounded) adversaries $\mathcal{A}$

$$\left| \Pr\left[1 \leftarrow \mathcal{A}(tr) \;\middle|\; \begin{array}{r} (\mathsf{x}, s) \leftarrow \mathcal{A}(1^\kappa) \\ tr \leftarrow \langle \mathcal{P}^*(\mathsf{x}, s), \mathcal{V}(\mathsf{x})\rangle \end{array} \right] - \Pr\left[1 \leftarrow \mathcal{A}(tr) \;\middle|\; \begin{array}{r} (\mathsf{x}, s) \leftarrow \mathcal{A}(1^\kappa) \\ (tr, \mathsf{w}) \leftarrow \mathcal{E}^{\langle \mathcal{P}^*(\mathsf{x},s), \mathcal{V}(\mathsf{x})\rangle}(u) \\ \text{if } tr \text{ is accepting then } (\mathsf{x}, \mathsf{w}) \in \mathcal{R} \end{array} \right] \right| \leq \mathsf{negl}(\kappa).$$

Definition 12. An $(n_1, \ldots, n_\mu)$-tree of accepting transcripts for the pair $\langle \mathcal{P}, \mathcal{V} \rangle$ with $2\mu+1$ rounds is a tree where:

- Each node of the tree in level i is labeled with the transcript of the protocol used up to $\mathcal{V}$'s i-th message.
- Each node in the same level i is labeled with a transcript that uses fresh (uniformly distributed and independent) randomness for the verifier's i-th challenges.
- Level i has n_i descendants.
- The leafs are labeled with transcripts that are accepted by the verifier.

Definition 13. The pair $\langle \mathcal{P}, \mathcal{V} \rangle$ has $(n_1, \ldots, n_\mu)$-generalized special soundness if there exists a PPT extractor $\mathcal{E}$ such that given an $(n_1, \ldots, n_\mu)$-tree of accepting transcripts for the pair $\langle \mathcal{P}, \mathcal{V} \rangle$, the extractor $\mathcal{E}$ outputs a valid witness for the statement.

Definition 14. An interactive proof system $\langle \mathcal{P}, \mathcal{V} \rangle$ is public coin if all messages from $\mathcal{V}$ to $\mathcal{P}$ are independent and uniformly distributed, and are uniquely defined by the randomness of the verifier alone.

Definition 15. A public coin interactive proof system $\langle \mathcal{P}, \mathcal{V} \rangle$ is perfect Honest Verifier Zero Knowledge (HVZK) if there exists a PPT simulator $\mathcal{S}$, such that for all PPT $\mathcal{A}$, it holds that

$$\Pr\left[1 \leftarrow \mathcal{A}(tr) \;\middle|\; (\mathsf{x}, \mathsf{w}, r) \leftarrow \mathcal{A}(1^\kappa) \wedge tr \leftarrow \langle \mathcal{P}^*(\mathsf{x}, \mathsf{w}), \mathcal{V}(\mathsf{x}; r)\rangle \wedge (\mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathcal{L}} \right] =$$
$$\Pr\left[1 \leftarrow \mathcal{A}(tr) \;\middle|\; (\mathsf{x}, \mathsf{w}, r) \leftarrow \mathcal{A}(1^\kappa) \wedge tr \leftarrow \mathcal{S}(\mathsf{x}, r) \wedge (\mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathcal{L}} \right].$$

Theorem 2. *Let $\langle \mathcal{P}, \mathcal{V} \rangle$ be a $2\mu+1$ round, public coin, interactive proof system with $(n_1, \ldots, n_\mu)$-generalized special soundness and $\prod_{i=1}^{\mu} n_i = O(\kappa^c)$ for a constant c. Then $\langle \mathcal{P}, \mathcal{V} \rangle$ has witness extended emulation.*

The proof of the theorem is given in [BCC+16]. Public coin interactive zero knowledge arguments can be transformed to non-interactive arguments in the random oracle model by means of the Fiat-Shamir transform.

2.7 Polynomial Commitment Schemes

In this section we define polynomial commitment schemes. Polynomial commitments were first introduced in [KZG10]. It is a cryptographic primitive that allows to a prover to give a short digest of a polynomial $p(X)$ and later convince a verifier that the evaluation of the polynomial committed polynomial $p(X)$ at a point z is v. The construction should be *binding*, meaning that after committing to $p(X)$, a prover cannot convince the verifier about two inconsistent evaluations, namely showing $p(v) = z$ and $p(v) = z'$ for $z \neq z'$.

We can also consider additional properties: a polynomial commitment scheme is *extractable* if we can "extract" a valid polynomial p from a prover that convinces a verifier with noticeable probability. Finally, regarding prover's privacy, we might require that the commitment is *hiding* if no information about the polynomial is leaked apart from the claimed evaluations.

We next give a formal definition of a polynomial commitment and the corresponding properties.

Definition 16 (Polynomial Commitment Scheme)**.** A polynomial commitment scheme PC is a tuple of PPT algorithms $(\mathsf{KeyGen}, \mathsf{Com}, \mathsf{Prove}, \mathsf{Verify})$ that work as follows:

$\mathsf{ck} \leftarrow \mathsf{PC.KeyGen}(1^\kappa, D)$**:** On input the security parameter κ and an upper bound on the degree D, it outputs a commitment key ck. The commitment key encodes a field $\mathbb{F}$ over which the polynomial is defined and a commitment space $\mathcal{C}$.

$(\mathsf{c}, \mathsf{aux}) \leftarrow \mathsf{PC.Com}(\mathsf{ck}, p)$**:** On input ck and a polynomial $p \in \mathbb{F}[X]$, it outputs a commitment $\mathsf{c} \in \mathcal{C}$ and opening information aux.

$\pi \leftarrow \mathsf{PC.Prove}(\mathsf{ck}, \mathsf{c}, p, \mathsf{aux}, d, z, v)$**:** On input ck, a commitment c, a polynomial p and opening information aux, a degree bound $d < D$ and a pair of values $z, v \in \mathbb{F}$, it outputs a proof π asserting that c is a commitment to a polynomial p of degree less than d and $p(z) = v$.

$0/1 \leftarrow \mathsf{PC.Verify}(\mathsf{ck}, \mathsf{c}, d, z, v, \pi)$**:** On input ck, a commitment c, a degree bound d, a pair of values $z, v \in \mathbb{F}$ and a proof π, it outputs a bit indicating acceptance or rejection of the claim.

that satisfies the following properties:

1. *Correctness.* For all $\kappa, D \in \mathbb{N}$, $p \in \mathbb{F}[X]$ of degree less than $d \leq D$ and $v, z \in \mathbb{F}$:

$$\Pr\left[\mathsf{PC.Verify}(\mathsf{ck}, \mathsf{c}, d, z, v, \pi) = 1 \;\middle|\; \begin{array}{c} \mathsf{ck} \leftarrow \mathsf{PC.KeyGen}(1^\kappa, D) \\ (\mathsf{c}, \mathsf{aux}) \leftarrow \mathsf{PC.Com}(\mathsf{ck}, p) \\ \pi \leftarrow \mathsf{PC.Prove}(\mathsf{ck}, \mathsf{c}, p, \mathsf{aux}, d, z, v) \end{array}\right] = 1$$

2. *Binding.* A polynomial commitment scheme is *binding* if, for all PPT adversaries $\mathcal{A}$, and all $\kappa, D \in \mathbb{N}$:

$$\Pr\left[\begin{array}{c}\mathsf{PC.Verify}(\mathsf{ck},\mathsf{c},d,z,v,\pi)=1\\ \mathsf{PC.Verify}(\mathsf{ck},\mathsf{c},d,z,v',\pi')=1\\ d<D,\ v\neq v'\end{array}\middle|\begin{array}{c}\mathsf{ck}\leftarrow\mathsf{PC.KeyGen}(1^\kappa,D)\\ (\mathsf{c},d,z,v,v',\pi,\pi')\leftarrow\mathcal{A}(\mathsf{ck},D)\end{array}\right]\leq\mathsf{negl}(\kappa)$$

3. *Succinctness.* A polynomial commitment succinct if the size of c and π are sublinear in the degree bound D.

We next define the extractability property of a polynomial commitment scheme.

Definition 17 (Extractable Polynomial Commitment Scheme)**.** A polynomial commitment scheme PC is extractable if for all $\kappa, D \in \mathbb{N}$ and all PPT adversaries $\mathcal{A}$, there exists an efficient PPT algorithm $\mathcal{E}$ such that

$$\Pr\left[\begin{array}{c}\mathsf{PC.Verify}(\mathsf{ck},\mathsf{c},d,z,v,\pi)=1\\ \Rightarrow\\ \mathsf{PC.Verify}(\mathsf{ck},\mathsf{c},d,z',v',\pi')=0\\ \vee\\ (p(z')=v'\wedge\deg_p\leq d)\end{array}\middle|\begin{array}{c}\mathsf{ck}\leftarrow\mathsf{PC.KeyGen}(1^\kappa,D)\\ (\mathsf{c},d,z,v,\pi)\leftarrow\mathcal{A}(\mathsf{ck},D)\\ p\leftarrow\mathcal{E}^{\mathcal{A}}(\mathsf{ck},D)\\ (z',v',\pi')\leftarrow\mathcal{A}(\mathsf{ck},D,p)\end{array}\right]\geq 1-\mathsf{negl}(\kappa)$$

The hiding notion is somewhat technical to describe. For a formal general definition, we refer the reader to [CHM+20]. For our needs, we will consider a simpler definition stating that (1) the commitment itself should be uniformly distributed over the commitment space C and (2) the open/verify algorithms are a zero knowledge argument for the language capturing valid commitment/opening pairs.

Definition 18 (Hiding Polynomial Commitment Scheme)**.** A polynomial commitment scheme PC is hiding if for all κ, D and $\mathsf{ck} \leftarrow \mathsf{PC.KeyGen}(1^\kappa, D)$:

1. For all $p \in \mathbb{F}[X]$ with degree less than D, the commitment c is uniformly distributed over C where $(\mathsf{c}, \mathsf{aux}) \leftarrow \mathsf{PC.Com}(\mathsf{ck}, p)$

2. The tuple (PC.KeyGen, PC.Prove, PC.Verify) is a non-interactive zero knowledge argument for the language

$$\mathcal{L} = \left\{\mathsf{c}, d, z, v \mid \exists p, \mathsf{aux}\ \text{s.t.}\ (\mathsf{c}, \mathsf{aux}) \leftarrow \mathsf{PC.Com}(\mathsf{ck}, p), \deg_p < d, p(z) = v\right\}$$

2.8 Delegation of Computation

Definition 19 (Delegation scheme in the preprocessing model)**.** A delegation scheme is a tuple of PPT algorithms Del = (Setup, Prove, Verify) such that

$\mathsf{srs} \leftarrow \mathsf{Setup}(1^\kappa, C)$: takes as input the security parameter κ and the description of an arithmetic circuit $C : \mathbb{F}^{n_0} \to \mathbb{F}^{n_d}$ and outputs an reference string srs.

$\pi \leftarrow \mathsf{Prove}(\mathsf{srs}, \mathsf{x}, \mathsf{y})$: takes as input the reference string srs, an input x and an output y, and outputs a proof π.

$0/1 \leftarrow \mathsf{Verify}(\mathsf{srs}, \mathsf{x}, \mathsf{y}, \pi)$: takes as input the reference string srs, an input x, an output y and a proof π, and outputs a bit indicating whether the proof is accepting or not.

that satisfies the following properties:

Completeness: For all n_0, n_d, all $C : \mathbb{F}^{n_0} \to \mathbb{F}^{n_d}$ and all x, y such that $\mathsf{y} = C(\mathsf{x})$ it holds that

$$\Pr\left[\mathsf{Verify}(\mathsf{srs}, \mathsf{x}, \mathsf{y}, \pi) = 1 \;\middle|\; \mathsf{srs} \leftarrow \mathsf{Setup}(1^\kappa, C), \pi \leftarrow \mathsf{Prove}(\mathsf{srs}, \mathsf{x}, \mathsf{y}) \right] \geq 1 - \mathsf{negl}(\kappa)$$

Soundness: For all $n_0, n_d \in \mathbb{N}$, all $C : \mathbb{F}^{n_0} \to \mathbb{F}^{n_d}$ and all PPT algorithms $\mathcal{A}$ it holds that

$$\Pr\left[\begin{array}{c} \mathsf{Verify}(\mathsf{srs}, \mathsf{x}, \mathsf{y}, \pi) = 1 \\ \mathsf{y} \neq C(\mathsf{x}) \end{array} \;\middle|\; \begin{array}{c} \mathsf{srs} \leftarrow \mathsf{Setup}(1^\kappa, C) \\ (\mathsf{x}, \mathsf{y}, \pi) \leftarrow \mathcal{A}(\mathsf{srs}) \end{array} \right] \leq \mathsf{negl}(\kappa)$$

Efficiency: The Setup and Prove algorithms run in time $\mathsf{poly}(|C|, \kappa)$. The size of the proof is $|\pi| = O(\kappa)$ and Verify algorithm runs in time $(n_0 + n_d)\mathsf{poly}(\kappa)$.

Chapter 3

Updateable Inner Product Argument with Logarithmic Verifier

This chapter is based on the paper "Updateable Inner Product Argument with Logarithmic Verifier and Applications" that was presented in PKC 2020, which is joint work with Vanesa Daza and Carla Ráfols.

Zero-Knowledge proofs have been an important primitive in the theory of cryptography since their introduction three decades ago. The classical applications of zero-knowledge proofs are numerous, including for example identification schemes, electronic voting, verifiable outsourced computation, or CCA secure public-key encryption. The common denominator of all of these is that zk-proofs are used to prove simple statements, like "this ciphertext is well-formed" or "I know a valid signature key". Although it was known that every NP statement could be proved in zero-knowledge [GMW87], the cost of such general proofs was prohibitive and more sophisticated applications of zk-proofs were completely impractical.

This situation has changed radically in the last decade with the introduction of pairing-based zk-SNARKs [Gro10]. The key element of these arguments is that they are *succinct*, i.e. sublinear in the size of the witness -in fact, they are *independent* of it- and thus very fast to verify. This is extremely powerful: in particular, a prover can show that it has executed correctly some large computation (expressed as a huge circuit) and a verifier will be convinced after doing only very few checks (e.g. computing 3 pairings in [Gro16]). Besides their scientific interest, SNARKs have opened the door to new real-world privacy-preserving applications. Cryptocurrencies like Zcash [BCG+14a] or Ethereum are two of the most popular examples so far.

However, even the most efficient instantiations of pairing-based SNARKs [Gro16; GM17] have a few drawbacks. On the efficiency side, the main ones are long common reference string and costly prover computation. On the security side, they are based on very strong hardness assumptions, and the setup is assumed to be trusted.

Recently, there are significant research efforts to propose alternatives which overcome some of these drawbacks following several dimensions. For instance, numerous works study how to reduce the trust in the common reference string, exploring weaker models such as subversion resistant SNARKs [BFS16; ABLZ17; Fuc18], updateable common reference strings [GKM+18] or transparent setup [BBHR19].

Moreover, there has been significant efforts in *abstracting* and *modularizing* the construction of SNARKs. A SNARK construction can be typically be described by three components: (1) an *arithmetization technique* that allows giving an algebraic essence in problems of combinatorial structure capturing the process of computation, such as Circuit Satisfiability, (2) a construction of a proof system in an information theoretic *idealized* model and (3) a *cryptographic compiler* that uses cryptographic tools to transform the idealized proof system in an actualal proof system that can be implemented in the real world.

This modular approach has numerous advantages. First, it makes SNARK constructions, which are usually complex and counterintuitive, easier to understand by breaking them down into simpler, independent components. Second, thanks to generic results, one can interchange components without the need to re-analyze the construction from scratch. Thus, a more efficient cryptographcic compilation for example can directly improve efficiency *of all* constructions that are constructed to work with this type of compiler.

By far the most common arithmetization technique is the rank one constraint satisfaction problem (R1CS) which generalizes the arithmetization used in [GGPR13] and essentially encodes an arithmetic circuit as a set of quadratic equations. An alternative arithmetization, which is used to encode *unifrom* (i.e. Turing machine based) computations is called algebraic intermediate representation [BBHR19]. The most commonly used idealized models are the interactive oracle proof model [BCS16] (IOP) and the algebraic holographic proof model [CHM+20] (AHP). In the former, the prover replies to verifier's challenges using long messages, which the verifier does not read in full; rather it queries a few randomly chosen positions of them to determine if the prover is honest or not. In the later, again the prover and verifier interact, but know the the messages of the prover are polynomials over a field; the verifier is allowed to query these polynomials at random points and learn their evaluations. A proof system in the IOP model is compiled to an actual proof system using a primitive called *vector commitment scheme* [CF13] and the latter using *polynomial commitments* [KZG10].

One of the most celebrated alternatives to pairing based SNARKs are the arguments of knowledge for Arithmetic Circuit Satisfiability of Bootle et al. [BCC+16] (and Bulletproofs, the improvement thereof by Bünz et al. [BBB+18]). The construction can be analyzed as a

Algebraic Holographic Proof compiled to a proof system using a novel polynomial commitment which is based on the *folding technique*. In fact, the primitive used is stronger in the sense that it implies a polynomial commitment scheme. The construction as a whole depends on weaker assumptions (the DLOG assumption and the Random Oracle if one wants to remove interaction via Fiat-Shamir). Furthermore, it *lacks the need for a trusted setup* and it features small proofs, logarithmic in the size of the witness. Unfortunately, verification time scales linearly, even when batching techniques are used. The motivation of this chapter is to demonstrate how to improve the cost of the verifier in the aforementioned works, while keeping most of its advantages.

Related Work

In [BCC+16], Bootle et al. proposed an interactive zero-knowledge argument at the heart of which lies a recursive argument for an inner product relation of committed values. The argument has very interesting properties, most notably it is transparent. The communication complexity is $O_\kappa(\log n)$ and the verification cost is $O_\kappa(n)$ where n is the size of the vectors. Prover complexity is asymptotically optimal ($O_\kappa(n)$) but it heavily uses expensive public-key operations. Bünz et al. [BBB+18] improved the communication complexity of the aforementioned protocol by a constant factor.

The inner product argument implies a polynomial commitment and as such, it has been employed in various SNARKs. The Muggle-proofs based [ZGK+17b; ZGK+17a; WTs+18; XZZ+19] proof systems that build on the delegation scheme of Goldwasser, Kalai, and Rothblum [GKR08], utilize it as such.

To mitigate the linear verification complexity of the construction a line of work has attempted to *amortize the verification cost* amongst various executions of the protocol. The crucial observation is that the linearity of the verifier depends on the system parameters and not on the specific instance proven. Starting from Halo [BGH19] and with various follow ups [BCMS20; BDFG21; KST21], not only the problem of amortizing verification cost of the inner product is achieved, but also other cryptographic primitives (PCD [BCCT13] and IVC [Val08]) are constructed based on the inner product argument.

An independent and concurrent work with the original publication this chapter is based on [BMM+21], generalizes the inner product argument in the pairing group setting. This is beneficial in two ways: first, the linearity of the verifier is mitigated by means of structured (instead of uniform) commitment keys and second, it allows to aggregate pairing product equations (instead of only equations on the field) which in turn allows useful aggregation techniques (see for example [GMN21; SCP+22]). The techniques we introduce in this chapter are similar to the former contribution of [BMM+21].

To achieve a middle ground between efficiency and trust, Groth et al. [GKM+18] defined the updateable model. In this model, everyone can non-interactively update the setup

parameters. As long as one update is honest, soundness is guaranteed. The authors also presented a scheme which is updateable, but it has a universal common reference string of size quadratic in the maximal size of all supported circuits (although from the global setup a linear, circuit-specific string can be derived). Maller et al. presented Sonic [MBKM19], which improved this to a linear srs by improving the reduction of [BCC+16]. Several works [GWC19; CHM+20; RZ21] have construct SNARKs in the universal and updateable models. However, all of these, including Sonic, are secure either in the Algebraic Group Model, or under knowledge type assumptions (apart from the Random Oracle Model). Finally, [BFS20] uses the techniques of the aforementioned results to construct a SNARK sound in groups of unknown order. When instantiated in class groups it achieves a transparent setup and asymptotically improves over STARKS [BBHR19] by a logarithmic factor.

Our Contribution

We construct an argument of knowledge for inner product relations on committed values in the updateable model. The construction is based on the folding technique of Bootle et al [BCC+16]. Its communication and verification complexity is *logarithmic* in the size of the statement. In contrast with the construction of Bootle et al, our inner product argument is not transparent. However, the complications of the srs generation are mitigated by the fact that the srs is updateable, as defined in [GKM+18]. Furthermore, existing structured reference strings from other applications can be used in the case of one of our instantiations, that is, the setup is *reusable*.

The inner product construction directly implies a polynomial commitment scheme, and hence, a universal succinct argument in the random oracle model [CHM+20]. As far as we know, all recently proposed efficient and fully succinct pairing based constructions [MBKM19; GWC19; CHM+20; RZ21] rely on the Algebraic Group Model [FKL18] or other knowledge type assumptions apart from the random oracle. This is because pairing based *extractable* polynomial commitments rely on such assumptions. In our case the random oracle model and a standard assumption is enough for an extractable polynomial commitment. The price we pay to remove such assumptions is settling for a logarithmic instead of constant proof/verification overhead. Compared to [BCC+16], we *exponentially* improve the verifier but we lose the transparent setup and we settle for the pairing group setting which is concretely more inefficient.

Our Techniques

Distribution Parameterized Vector Commitments. We revisit the use of vector commitment schemes in zero-knowledge proof systems when working in groups: instead of using the classical Pedersen commitment key which is uniformly sampled, we add some limited structure which simultaneously allows more efficient representation of the key

and efficient updateability. When combined with the properties of bilinear groups, only a compressed version of it is enough to allow a verifier to perform verification tasks exponentially faster.

In particular we propose two instantiations:

- The commitment key consisting of group encodings of all monomials of a secret x, i.e., $[1], [x], [x^2], \ldots, [x^{n-1}]$.
- The commitment key consisting of group encodings of all multilinear monomials monomials of a secret $x_1, \ldots, x_v$ i.e. $[1], [x_1], [x_2], [x_1 x_2], \ldots, [x_1 x_2 \cdots x_v]$.

The structure of both commitment keys allows to non-interactively update the parameters and thus nullifying the trapdoors x or $x_1, \ldots, x_n$. We take advantage of this structure in bilinear groups to create compressed versions of these keys of size only $\log n$. For various languages, this allows the verifier to verify statements with the help of the prover without reading the whole commitment key. This leads to exponentially faster verification of proofs with minimal overhead for the prover, at the price of moving to bilinear instead of plain DLOG groups. The former distribution requires a larger ($O(n)$ size) assumption but one can find such srss "on the wild" since they are already used. The latter reduces to a constant size assumption but -to the best of our knowledge- their use in not that much spread as far as real world applications are concerned.

Inner Program Argument with Logarithmic Verifier. Using these techniques, we modify the inner product protocol of Bootle et al. [BCC+16] for proving that for given commitments $c_1 = \mathsf{Com}(\mathbf{a})$, $c_2 = \mathsf{Com}(\mathbf{b})$ and $z \in \mathbb{F}$, it holds that $\mathbf{a}^\top \mathbf{b} = z$. More specifically, we note that the overhead of the verifier in [BCC+16] is computing a new commitment key in each of the $\log n$ rounds of the protocol, where n is the vector dimension. This key depends on the previous key and the verifiers' challenges. Instead of doing that, we only give the verifier the compressed key (which is logarithmic in n) and have the prover convince the verifier that the reduced statement is w.r.t. a new key which is the correct one.

Polynomial Commitment Scheme. Finally, we construct a polynomial commitment scheme based on the aforementioned argument. We do this without requiring any additional assumptions, we only need to add some more rounds of interaction between the prover and the verifier. While a polynomial commitment is almost immediately derived from an inner product argument (the polynomial evaluation relation can in fact be written as an inner product relation) we need to consider some technicalities, specifically including a low degree test on the polynomials and augment them with privacy properties (note that the inner product argument is not in fact zero knowledge, although it can be augmented to a zero knowledge one with relatively minor modifications). The polynomial commitment scheme implies universal zero knowledge SNARKs in the random oracle model without additional assumptions, by means of generic transformations, see for example [CHM+20].

3.1 Distribution Parameterized Pedersen Commitment Scheme

In this section we first define an updateability property of a commitment scheme, similarly to the way it is defined for SNARKs in [GKM+18]. Essentially, this property states that we can *resample* the commitment key without (1) skewing its distribution and (2) invalidating the already included randomness. This is usefull since it allows performing a cheaper trusted setup. Each party *updates* the commitment key in a non-interactive fashion and it is enough to believe a single party is honest to trust the parameters. The latter conditions holds since we know that the randomness of the honest party is not invalidated, hence, binding property should hold.

Next we introduce a generalization of Pedersen commitment which is updateable and can be succinctly represented. This is the core ingredient in reducing the verification time of the folding technique. Essentially, we do not sample uniformly distributed keys as is the case in Pedersen commitment, but we sample from any matrix distribution that satisfies the FindRep assumption. Depending on the intended use, we can simply use the apporpriate distribution.

3.1.1 Updateable Commitment Schemes

We define commitment schemes which have an updateability property as well. We do this to simplify proofs in the following sections. An updateable commitment will be enough to guarantee updateability of all the protocols in this work, since all the arguments presented hold regardless of parameters *unless* there is a breach in the binding property of the commitment scheme.

Definition 20. An Updateable Commitment Scheme is a tuple of algorithms $\mathsf{CS} = (\mathsf{Setup}, \mathsf{VerSetup}, \mathsf{Update}, \mathsf{VerUpdate}, \mathsf{Com}, \mathsf{Open})$ such that

- $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\kappa)$: takes as input the security parameter κ and outputs a commitment key ck.
- $(\mathsf{ck}', \pi_{\mathsf{ck}'}) \leftarrow \mathsf{Update}(\mathsf{ck})$: takes as input a commitment key ck and produces a new commitment key ck' and a proof of correct update $\pi_{\mathsf{ck}'}$.
- $0/1 \leftarrow \mathsf{VerSetup}(\mathsf{ck}, 1^\kappa, n)$: takes as input a commitment key ck and the security parameter κ, and outputs a bit indicating the correctness of the structure of the key.
- $0/1 \leftarrow \mathsf{VerUpdate}(\mathsf{ck}', \mathsf{ck}, \pi_{\mathsf{ck}'})$: takes as input a new key ck', an old key ck and a proof $\pi_{\mathsf{ck}'}$, and outputs a bit indicating update correctness.
- $(\mathsf{c}, \pi) \leftarrow \mathsf{Com}(\mathsf{ck}, \mathsf{m})$: takes as input the commitment key and a message $\mathsf{m} \in \mathcal{M}$, and outputs a commitment $\mathsf{c} \in C$ and an opening proof π.
- $0/1 \leftarrow \mathsf{Open}(\mathsf{ck}, \mathsf{c}, \mathsf{m}, \pi)$: takes as input the commitment key ck, the commitment c,

the message m, and the opening proof π and outputs a bit indicating the validity of the opening.

which is correct, updateable binding and (optionally) hiding as defined next.

Definition 21. An Updateable Commitment Scheme is correct if

1. for all κ
$$\Pr\left[\ \mathsf{VerSetup}(\mathsf{ck}, 1^\kappa) = 1 \ \middle|\ \mathsf{ck} \leftarrow \mathsf{Setup}(1^\kappa)\ \right] = 1,$$
2. for all κ, ck
$$\Pr\left[\begin{array}{c}\mathsf{VerSetup}(\mathsf{ck}', 1^\kappa) = 1 \wedge \\ \mathsf{VerUpdate}(\mathsf{ck}, \mathsf{ck}', \pi_{\mathsf{ck}'}) = 1\end{array}\ \middle|\ \begin{array}{c}\mathsf{VerSetup}(\mathsf{ck}, 1^\kappa) = 1 \wedge \\ (\mathsf{ck}', \pi_{\mathsf{ck}'}) \leftarrow \mathsf{Update}(\mathsf{ck})\end{array}\right] = 1$$
3. and for all $\kappa, \mathsf{ck}, \mathsf{m}$
$$\Pr\left[\ \mathsf{Open}(\mathsf{ck}, \mathsf{c}, \mathsf{m}, \pi) = 1\ \middle|\ \begin{array}{c}\mathsf{VerSetup}(\mathsf{ck}, 1^\kappa, n) = 1 \wedge \\ (\mathsf{c}, \pi) \leftarrow \mathsf{Com}(\mathsf{ck}, \mathsf{m})\end{array}\right] = 1.$$

Definition 22. An Updateable Commitment Scheme has the Updateable Computational Binding property if for all stateful PPT $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, and for all κ,

$$\Pr\left[\begin{array}{c}\mathsf{VerSetup}(\mathsf{ck}_1, 1^\kappa) = 1 \\ \mathsf{VerUpdate}(\mathsf{ck}_3, \mathsf{ck}_2, \pi_{\mathsf{ck}_3}) = 1 \\ \mathsf{Open}(\mathsf{ck}_3, \mathsf{c}, \mathsf{m}, \pi_1) = 1 \\ \mathsf{Open}(\mathsf{ck}_3, \mathsf{c}, \mathsf{m}, \pi_2) = 1 \\ \mathsf{m}_1 \neq \mathsf{m}_2\end{array}\ \middle|\ \begin{array}{r}(\mathsf{ck}_1, \mathtt{st}_1) \leftarrow \mathcal{A}_1(1^\kappa) \\ (\mathsf{ck}_2, \pi_{\mathsf{ck}_2}) \leftarrow \mathsf{Update}(\mathsf{ck}_1) \\ (\mathsf{ck}_3, \pi_{\mathsf{ck}_3}, \mathtt{st}_2) \leftarrow \mathcal{A}_2(\mathtt{st}_1, \mathsf{ck}_2, \pi_{\mathsf{ck}_2}) \wedge \\ (\mathsf{c}, \mathsf{m}_1, \pi_1, \mathsf{m}_2, \pi_2) \leftarrow \mathcal{A}_3(\mathtt{st}_2)\end{array}\right] \leq \mathsf{negl}(\kappa).$$

Definition 23. An Updateable Commitment Scheme is perfectly hiding if, for all κ, m, and all ck s.t. $\mathsf{VerSetup}(\mathsf{ck}, 1^\kappa, n) = 1$, and all c_1

$$\Pr\left[\ \mathsf{c} = \mathsf{c}_1\ \middle|\ (\mathsf{c}, \pi) \leftarrow \mathsf{Com}(\mathsf{ck}, \mathsf{m})\ \right] = \Pr\left[\ \mathsf{c} = \mathsf{c}_1\ \middle|\ \mathsf{c} \leftarrow \mathcal{C}\ \right].$$

3.1.2 Construction

In this section we show how to construct commitment schemes under any distribution $\mathcal{D}_{1,n}$ that satisfies the $\mathcal{D}_{1,n}$-FindRep assumption. We specifically consider the univariate and multilinear distributions $\mathcal{X}_{1,1,n}$, $\mathcal{X}_{\nu,1,1}$, distributions which also have the updateability property (note that for the uniform distribution, i.e. the Pedersen Vector Commitment, updateability trivially holds since the Setup is transparent). For the remainder of the section we refer to them as $\mathcal{X}_n$ and $\mathcal{X}_{\nu,1}$ respectively.

We first present the generic construction in Fig 3.1 and then discuss the updateability properties of the two specific distributions. The commitment scheme is canonical so we only present the Setup and Com algorithms.

Figure 3.1 Distribution Parameterized Pedersen Commitment Scheme. The scheme is parameterized by a matrix distribution $\mathcal{D}_{1,n}$.

$\mathsf{Setup}(1^\kappa)$**:**

$\mathsf{gk} \leftarrow \mathcal{G}(1^\kappa)$

$[\mathbf{r}] \leftarrow \mathcal{D}_{1,n}$, $[\rho] \leftarrow \mathbb{G}_1$

Output $[\mathbf{r}], [\rho]$

$\mathsf{Com}([\mathbf{r}], \mathbf{m})$**:**

$\rho \leftarrow \mathbb{F}$.

Output $[\mathsf{c}]_1 = ([\mathbf{r}]^\top \ [\rho]) \begin{pmatrix} \mathbf{m} \\ \rho \end{pmatrix}$, $\pi = \rho$

Theorem 3. *The commitment scheme of Fig. 3.1 is a commitment scheme that is binding under the* $\mathcal{D}_{1,n}$-FindRep *assumption and unconditionally hiding.*

Proof. Correctness follows by inspection and hiding follows since commitments are uniformly distributed group elements…We next show it is binding.

Consider an adversary $\mathcal{A}$ outputting a commitment c and two valid openings $\mathbf{m}_1, \rho_1$ and $\mathbf{m}_2, \rho_2$ with $\mathbf{m} \neq \mathbf{m}'$. Then, it should be the case that

$$([\mathbf{r}]^\top \ [\rho]) \begin{pmatrix} \mathbf{m}_1 \\ \rho_1 \end{pmatrix} = [\mathsf{c}]_1 = ([\mathbf{r}]^\top \ [\rho]) \begin{pmatrix} \mathbf{m}_2 \\ \rho_2 \end{pmatrix} \Leftrightarrow ([\mathbf{r}]^\top \ [\rho]) \begin{pmatrix} \mathbf{m}_1 - \mathbf{m}_2 \\ \rho_1 - \rho_2 \end{pmatrix} = [0]$$

which is a non-trivial solution to the $\mathcal{D}'_{1,n}$-FindRep, where $\mathcal{D}'$ is the same as $\mathcal{D}$ but also outputs an extra uniformly distributed element. We finally show that solving $\mathcal{D}'_{1,n}$-FindRep implies sovling $\mathcal{D}'_{1,n}$-FindRep.

Let $\mathbf{r}$ be a challenge sampled according to $\mathcal{D}_{1,n}$. We then set $[\rho]_1 = [r_1]_1\tau$ for a uniform $\tau \leftarrow \mathbb{F}$. Let $\mathcal{A}$ be an adversary against $\mathcal{D}'_{1,n}$. Assume it successfully outputs $\mathbf{w} \neq 0$ such that $(\mathbf{r}^\top \ \rho)\mathbf{w} = 0$. Writing $\mathbf{w} = (\mathbf{w}', z)$ we get $\mathbf{r}^\top\mathbf{w}' + r_1\tau z = 0$ and $\mathbf{w}' + \tau z\mathbf{e}_1$ is an element in the kernel of $\mathbf{r}$. Either this element is non-zero, in which case we are done, or it is the zero vector. In the latter case, we have $w_1 = \tau z$, which means that $\tau = w_1 z^{-1}$. The latter is a solution to a DLOG instance for the elements $[r_1]$ and $[\rho]$ which is a weaker assumption. □

We next present the multilinear case ($\mathcal{X}_{\nu,1}$) case in detail and discuss which modifications are needed for the monomial case $\mathcal{X}_n$. We introduce some notation first.

Notation. Let $\mathbf{x} = (x_1, \ldots, x_\nu) \in \mathbb{F}^n$. We denote with $\overline{\mathbf{x}}$ the vector of multilinear polynomials evaluated at $\mathbf{x}$. We define it recursively, namely, $\overline{\mathbf{x}}_\nu = (\overline{\mathbf{x}}_{\nu-1}, x_\nu\overline{\mathbf{x}}_{\nu-1})$ where $\overline{\mathbf{x}}_0 = (1)$. We also denote $\mathbf{x}^\mathbf{n} = (1, x, \ldots, x^{n-1})$.

Figure 3.2 Updateable instantiation of distribution parameterized Pedersen commitment scheme using the $\mathcal{X}_{\nu,1}$ distribution. We assume NIZK is a non-interactive argument for the discrete logarithm language

$\mathsf{VerSetup}([\mathbf{r}]_1(=[\overline{\mathbf{x}}]_1),[\mathbf{x}]_2)$: :

Verify $[r_1]_1 = [1]_1$

For $1 \leq i \leq \nu$ and $1 \leq j \leq 2^{i-1}$, verify $e\left([r_{2^{i-1}+j}]_1,[1]_2\right) = e\left([r_j]_1,[x_i]_2\right)$

$\mathsf{Update}([\mathbf{r}]_1(=[\overline{\mathbf{x}}]_1),[\mathbf{x}]_2)$:

$\mathbf{y} \leftarrow \mathbb{F}^{\nu}$.

Compute $[\mathbf{r}']_1 \leftarrow \overline{\mathbf{y}} \circ [\mathbf{r}]_1$, $[\mathbf{x}']_2 \leftarrow \mathbf{y} \circ [\mathbf{x}]_2$

For $1 \leq i \leq \nu$: $\pi_i \leftarrow \mathsf{NIZK.Prove}(\mathsf{srs}, \mathsf{x} = ([x_i]_2,[x'_i]_2), \mathsf{w} = y_i)$

$\mathsf{VerUpdate}([\mathbf{x}]_2,[\mathbf{x}']_2,[\mathbf{r}']_1,\pi_1,\ldots,\pi_\nu)$:

For $1 \leq i \leq \nu$: $b_i \leftarrow \mathsf{NIZK.Verify}(\mathsf{srs}, \mathsf{x} = ([x_i]_2,[x'_i]_2), \pi_i)$

$b \leftarrow \mathsf{VerSetup}(\mathsf{par},[\mathbf{r}]_1,[\mathbf{x}]_2)$

Output $b \wedge b_1 \wedge b_\nu$

For our application it is sufficient to give in $\mathbb{G}_2$ only the elements that define the commitment key, and not the whole key vector, 0 i.e. $[\mathbf{x}]_2$ such that $\mathbf{r} = \overline{\mathbf{r}}$. Looking ahead, in the inner product argument $[\mathbf{x}]_2$ will be the compressed key the verifier has.

The update mechanism is fairly simple. To check a commitment key's structure, simply assert the various DDH relations that are implied by the $\mathcal{X}_{\nu,1}$ distribution, and to update, sample a vector from the same distribution and multiply it pairwise with the current key. NIZK PoK are used to assert that the previous randomness is taken into account in the new key and to ensure that any party updating knows its contribution to the final commitment key. We omit the Setup and Com algorithms from the description since they are the same as the generic construction. We present the construction in Fig. 3.2.

Theorem 4. *Let* NIZK *be a NIZK for the relation* $\mathcal{R} = \{(([x],[x']),y) : [x'] = y[x]\}$. *Then, the Pedersen commitment scheme parameterized by the* $\mathcal{X}_{\nu,1}$ *distribution is Updateably Computationally Binding under the* $\mathcal{X}_{\nu,1}$-FindRep *assumption.*

Proof. We work as follows: given an adversary $\mathcal{A}$ in the scenario that breaks the binding property, we construct an adversary $\mathcal{A}'$ that breaks the $\mathcal{X}_{\nu,1}$-FindRep assumption. $\mathcal{A}'$ acts as follows:

- $\mathcal{A}'$ creates parameters for the NIZK.
- On input $[\overline{\mathbf{x}}]_{1,2}$ it invokes $\mathcal{A}$, gets parameters $[\mathbf{s}]_1,[\mathbf{y}]_2$ and uses VerSetup to verify them.

- It performs an update that supposedly results in the parameters it received. It uses the nizk simulator to construct convincing proofs. Specifically, it computes simulated proofs
$$\pi_i \leftarrow \mathsf{NIZK.Prove}(\mathsf{srs}, \mathsf{x} = [y_i]_2, [x_i]_2)$$
and gives $[\overline{\mathbf{x}}]_1, [\mathbf{x}]_2, \pi_1, \ldots, \pi_\nu$ to $\mathcal{A}$.
- $\mathcal{A}$ responds with its own update, namely parameters $[\mathbf{t}]_1, [\mathbf{z}]_2$ and proofs $\pi'_1, \ldots, \pi'_\nu$ that assert the relations $[z_i]_2 = x'_i[x_i]_2$. It also gives a a commitment $[\mathsf{c}]_1$ and two valid openings $\mathbf{a}_1 \neq \mathbf{a}_2$ for these parameters.
- $\mathcal{A}'$ runs the knowledge extractor to get $x'_1, \ldots, x'_\nu$.
- It outputs $[\mathsf{c}]_1, \mathbf{a}_1 \circ \overline{\mathbf{x}'}, \mathbf{a_2} \circ \overline{\mathbf{x}'}$.

First note that the NIZK simulator implies that the view of $\mathcal{A}$ is identical (assuming the NIZK AoK to have perfect zero knowledge) to that in a real experiment. Assuming the adversary is successful we have that for $\mathbf{a}_1 \neq \mathbf{a}_2$,

$$\mathbf{a}_1^\top \mathbf{t} = \mathbf{a}_2^\top \mathbf{t} \Leftrightarrow \mathbf{a}_1^\top(\overline{\mathbf{x}'} \circ \overline{\mathbf{x}}) = \mathbf{a}_2^\top(\overline{\mathbf{x}'} \circ \overline{\mathbf{x}}) \Leftrightarrow (\mathbf{a}_1 \circ \overline{\mathbf{x}'})^\top \overline{\mathbf{x}} = (\mathbf{a}_2 \circ \overline{\mathbf{x}'})^\top \overline{\mathbf{x}},$$

which consists a binding attack with commitment key $\overline{\mathbf{x}}$. We then break the $\mathcal{X}_{\nu,1}$-FindRep assumption as in proof of Thm. 3. □

We can use a transparent scheme such as [BBB+18] to prove that an update is correctly performed, which will yield $O(\log \log n)$ proof size.

A similar construction works for the $\mathcal{X}_n$ distribution. In this case, we simply need the element x encoded in $\mathbb{G}_2$ since this is enough to check that the key is drawn from the correct distribution. That is, for each i, it is enough to check that $e([r]_1, [1]_2) = e([r_{i-1}]_1, [x]_2)$. The Update and VerUpdate work in the same way but now a NIZK AoK is only needed for the element $[x]_2$.

As for concrete efficiency, the cost is dominated by the group exponentiations and the pairing operations for the verifier (the NIZK AoK statements are logarithmic in n). Setup and Update are dominated by n exponentiations in $\mathbb{G}_1$, VerSetup and VerUpdate by n pairing operations, and Com and Open by one multi-exponentiation of size n in $\mathbb{G}_1$ which, if performed trivially needs n exponentiations. Proof size amounts to $\log n$ proofs of the NIZK AoK in the $\mathcal{X}_{\nu,1}$ case and 1 in the $\mathcal{X}_n$ case.

3.2 Improved Inner Product Argument

In this section, we will first provide a high-level description of the inner product argument of [BCC+16], which has linear verification cost. Next, we briefly discuss how to reduce the

verification complexity to logarithmic in the designated verifier setting in the srs model by changing the distribution of the commitment keys. Finally, we "compile" the designated verifier construction in a public verifiable one in the pairing group setting.

We first briefly present the Inner Product Argument of [BCC+16]. The argument is a Proof of Knowledge of the openings of two (non-hiding) Vector Pedersen Commitments that satisfy an inner product relation. In [BCC+16], keys are sampled from $\mathcal{U}_{1,n}$. Formally, it is a proof of knowledge for the following language $\mathcal{L}_{\mathsf{IP}}$:

$$
\begin{aligned}
&([\mathbf{r}],[\mathbf{s}] \in \mathbb{G}^{2^\nu}, [\alpha], [\beta] \in \mathbb{G}, z \in \mathbb{F}) \in \mathcal{L}_{\mathsf{IP}} \iff \\
&\quad \exists \mathbf{a}, \mathbf{b} \in \mathbb{F}^{2^\nu} \text{ s.t. } [\alpha] = [\mathbf{r}^\top \mathbf{a}] \wedge [\beta] = [\mathbf{s}^\top \mathbf{b}] \wedge \mathbf{a}^\top \mathbf{b} = z.
\end{aligned}
$$

The idea of the protocol is to reduce this statement to an equivalent one of roughly half the size.

To do that, we create new commitment keys which have size half of the original one by splitting them in half and then combining them to a new key based on a challenge issued by the verifier. That is, the new commitment key will be $[\mathbf{r}'] = c^{-1}[\mathbf{r}_0] + c^{-2}[\mathbf{r}_1]$, where c is the verifier's challenge.

In order to prevent the prover from taking advantage of the split, we first ask her to give partial commitments $[\alpha_{-1}] = [\mathbf{r}_1^\top \mathbf{a}_0]$, $[\alpha_1] = [\mathbf{r}_0^\top \mathbf{a}_1]$,

The new witness will be $\mathbf{a}' = c\mathbf{a}_0 + c^2\mathbf{a}_1$. Note that both prover and verifier can compute the commitment to this new value, for every challenge c, from the partial commitments as follows:

$$
\begin{aligned}
[\alpha'] = [\mathbf{r}'^\top \mathbf{a}'] &= [(c^{-1}\mathbf{r}_0 + c^{-2}\mathbf{r}_1)(c\mathbf{a}_0 + c^2\mathbf{a}_1)] \\
&= [\mathbf{r}_0^\top \mathbf{a}_0] + [\mathbf{r}_1^\top \mathbf{a}_1] + [c^{-1}\mathbf{r}_1\mathbf{a}_0] + [c\mathbf{r}_0\mathbf{a}_1] \\
&= [\alpha] + c^{-1}[\alpha_{-1}] + c[\alpha_1]
\end{aligned}
$$

The same procedure is done for the second commitment $[\beta] = [\mathbf{s}^\top \mathbf{b}]$ with the inverse challenge c^{-1}.

Finally, the prover sends before seeing the challenge c the cross term values $z_{-1} = \mathbf{a}_1^\top \mathbf{b}_0$ and $z_1 = \mathbf{a}_0^\top \mathbf{b}_1$, and based on these, the new inner product is computed as $z' = z_{-1}c + z + z_1 c^{-1}$. The new statement becomes $([\mathbf{r}'], [\mathbf{s}'], [\alpha'], [\beta'], z') \in \mathcal{L}_{\mathsf{IP}}$.

Straightforward calculations assert that the new witness is indeed a witness for the new statement. The prover can now simply send the new witness $\mathbf{a}', \mathbf{b}'$ with cost half of what it would take to send $\mathbf{a}, \mathbf{b}$.

Figure 3.3 Recursive step of the inner product argument of [BCC+16]. When the statement size is small (constant), the prover simply sends the witness to the verifier.

$$\mathsf{x} = ([\mathbf{r}], [\mathbf{s}], [\alpha], [\beta], z),\ \mathsf{w} = (\mathbf{a}, \mathbf{b}),$$
$$\text{claim: } \mathsf{x} \in \mathcal{L}_{\mathsf{IP}}$$

$\mathcal{P}: \mathsf{x}, \mathsf{w}$		$\mathcal{V}: \mathsf{x}$
$[\alpha_{-1}] = [\mathbf{r}_1]^\top \mathbf{a}_0,\ [\alpha_1] = [\mathbf{r}_0]^\top \mathbf{a}_1$ $[\beta_{-1}] = [\mathbf{s}_1]^\top \mathbf{b}_0,\ [\beta_1] = [\mathbf{s}_0]^\top \mathbf{b}_1$		
$z_{-1} = \mathbf{a}_1^\top \mathbf{b}_0,\ z_1 = \mathbf{a}_0^\top \mathbf{b}_1,$	$\xrightarrow{[\alpha_{-1}], [\alpha_1], [\beta_{-1}], [\beta_1], z_{-1}, z_1}$	
	$\xleftarrow{c}$	$c \leftarrow \mathbb{F}$
$\mathbf{a}' = c\mathbf{a}_0 + c^2 \chi \mathbf{a}_1$ $\mathbf{b}' = c^{-1}\mathbf{b}_0 + c^{-2}\chi \mathbf{b}_1$		$[\mathbf{r}'] = c_{-1}[\mathbf{r}_0] + c_{-2}[\mathbf{r}_1]$ $[\mathbf{s}'] = c[\mathbf{s}_0] + c_2[\mathbf{s}_1]$ $[\alpha'] = c^{-1}[\alpha_{-1}] + [\alpha] + c[\alpha_1]$ $[\beta'] = c[\beta_{-1}] + [\beta] + c^{-1}[\beta_1]$ $z' = z_{-1}c + z + z_1 c^{-1}$
$\mathsf{w}' = (\mathbf{a}', \mathbf{b}')$		$\mathsf{x}' = ([\mathbf{r}'], [\mathbf{s}'], [\alpha'], [\beta'], z')$

Designated Verifiability with Logarithmic Complexity. We next discuss how to modify the above protocol with a $\mathcal{D}$-variant of the commitment scheme to achieve a logarithmic verifier. Full details are only given for the public verifiable scheme, which is very similar.

The linear overhead in the verifier's computation is computing the new keys $\mathbf{r}', \mathbf{s}'$. Having a structured commitment key allows to make this computation implicit for the verifier. Consider the first key $\mathbf{r}$. If $\mathbf{r} \leftarrow \mathcal{X}_{\nu,1}$, then $\mathbf{r} = (\mathbf{r}_0, \mathbf{r}_1) = (\mathbf{r}_0, x_\nu \mathbf{r}_0)$. So, in the first round, the key for the next round is

$$[\mathbf{r}'] = c^{-1}[\mathbf{r}_0] + c^{-2}[\mathbf{r}_1] = (c^{-1} + x_\nu c^{-2})[\mathbf{r}_0]$$

The new key is now determined by $[x_1], \ldots, [x_{\nu-1}]$ and the new generator is $(c^{-1} + x_\nu c^{-2})[1]$. Further, this transformation respects the structure of the key, which can again be written as $\mathbf{r}' = (\mathbf{r}'_0, x_{\nu-1}\mathbf{r}'_0)$, so the same argument can be applied again.

In the designated verifier case, we let the verifier know $x_1, \ldots, x_\nu$. It does not compute or read $[\mathbf{r}']$ in each round but just checks in the last round if:

$$[r'] = \prod_{i=1}^{\nu} (c_i^{-1} + x_{\nu-i+1} c_i^{-2})[1]$$

where c_i is the challenge at round i, and $[r']$ is the key in the last round (consisting of 1 element). The same holds for the second key $[s']$. Therefore, verification requires a logarithmic number of operations.

When $\mathbf{r} \leftarrow \mathcal{X}_{2^\nu}$, the verification can also be reduced to logarithmic, as the structure of the key is very similar, namely, $\mathbf{r} = (\mathbf{r}_1, \mathbf{r}_1) = (\mathbf{r}_0, x^{2^{\nu-1}}\mathbf{r}_0)$. The $\mathcal{X}_{2^\nu}$ distribution is in fact a special case of the $\mathcal{X}_{\nu,1}$ distribution where $x_i = x^{2^{i-1}}$, so the same technique applies.

Public Verifiability with Logarithmic Complexity. To allow public verifiability, we work in the pairing group setting. The verifier can no longer compute

$$\prod_{i=1}^{\nu}(c_i^{-1} + x_{\nu-i+1}c_i^{-2})[1]$$

but it is easy to observe that it actually does not need to do so. It simply needs to verify this relation which it can do with the help of the prover and by using the pairing operation as a DDH oracle.

We next present the argument formally for the $\mathcal{X}_{\nu,1}$ distribution (the univariate distribution is a special case). First, we define the language of well structured commitments. We include the generator since it will be modified in each round.

$$([r]_1, [\mathbf{r}]_1, [\mathbf{x}]_2) \in \mathcal{L}_{\mathsf{Com}}^{\mathcal{X}_{\nu,1}} \iff$$
$$[r_1]_1 = [r] \wedge \forall i \in \{1, \ldots, \nu\}\, \forall j \in \left\{1, \ldots, 2^{i-1}\right\} [r_{2^{i-1}+j}]_1 = x_i[r_j]_1$$

We next modify to language $\mathcal{L}_{\mathsf{IP}}$ to capture the extra requirements regarding the distribution of the commitment keys. We describe the new protocol in Fig. 3.4.

$$\begin{aligned}
&([r]_1, [s]_1, [\mathbf{x}]_2, [\mathbf{y}]_2, [\alpha]_1, [\beta]_1, z) \in \mathcal{L}_{\mathsf{IP}} \iff \\
&\quad \exists\, [\mathbf{r}]_1, [\mathbf{s}]_1 \in \mathbb{G}_1^{2^\nu}, \mathbf{a}, \mathbf{b} \in \mathbb{F}^{2^\nu} \text{ s.t.} \\
&\quad ([r]_1, [\mathbf{r}]_1, [\mathbf{x}]_2) \in \mathcal{L}_{\mathsf{Com}}^{\mathcal{X}_{\nu,1}} \wedge ([s]_1, [\mathbf{s}]_1, [\mathbf{y}]_2) \in \mathcal{L}_{\mathsf{Com}}^{\mathcal{X}_{\nu,1}} \wedge \\
&\quad [\alpha]_1 = [\mathbf{a}^\top \mathbf{r}]_1 \wedge [\beta]_1 = [\mathbf{b}^\top \mathbf{s}]_1 \wedge \mathbf{a}^\top \mathbf{b} = z.
\end{aligned}$$

Theorem 5. *Protocol of Fig. 3.4 is a public coin, argument of knowledge for the relation $\mathcal{L}_{\mathsf{IP}}$ with ν round complexity, $O(2^\nu)$ prover complexity, and $O(\nu)$ communication and verification complexity under either the $\mathcal{X}_{\nu,1}$-FindRep assumption (or the $\mathcal{X}_{2^\nu}$-FindRep assumption). The argument yields an updateable non-interactive argument of knowledge in the random oracle model. In the former case the proof size of an update is $O(\nu)$ and in the latter $O(1)$.*

Proof.
Completeness: We show that each reduction round leads to a valid reduced statement. It is enough to show that the prover and verifier compute the same key. Then, we can argue as in the case with uniform keys.

First, note that $[\mathbf{r}']_1 = c^{-1}[\mathbf{r_0}]_1 + c^{-2}[\mathbf{r_1}]_1$, which means that we "combine" all pair of elements that have distance $2^{\nu-1}$. That is, for all $j \leq 2^{\nu-1}$,

$$[r'_j]_1 = c^{-1}[r_j]_1 + c^{-2}[r_{2^{\nu-1}+j}]_1$$

Figure 3.4 Recursive step of the improved inner product argument. When the statement size is small (constant), the prover simply sends the witness to the verifier.

$\mathsf{x} = ([r]_1, [s]_1, [\mathbf{x}]_2, [\mathbf{y}]_2, [\alpha]_1, [\beta]_1, z), \mathsf{w} = ([\mathbf{r}]_1, [\mathbf{s}]_1, \mathbf{a}, \mathbf{b})$,
claim: $\mathsf{x} \in \mathcal{L}_{\mathsf{IP}}$

$\mathcal{P} : \mathsf{x}, \mathsf{w}$		$\mathcal{V} : \mathsf{x}$
$[\alpha_{-1}]_1 = [\mathbf{r}_1]_1^\top \mathbf{a}_0$ $[\alpha_1]_1 = [\mathbf{r}_0]_1^\top \mathbf{a}_1$ $[\beta_{-1}]_1 = [\mathbf{s}_1]_1^\top \mathbf{b}_0$ $[\beta_1]_1 = [\mathbf{s}_0]_1^\top \mathbf{b}_1$		
$z_{-1} = \mathbf{a}_1^\top \mathbf{b}_0,\ z_1 = \mathbf{a}_0^\top \mathbf{b}_1,$	$\xrightarrow{[\alpha_{-1}]_1, [\alpha_1]_1, [\beta_{-1}]_1, [\beta_1]_1, z_{-1}, z_1}$	
	$\xleftarrow{c}$	$c \leftarrow \mathbb{F}$
$[\mathbf{r}']_1 = c^{-1}[\mathbf{r}_0]_1 + c^{-2}[\mathbf{r}_1]_1$ $[\mathbf{s}']_1 = c[\mathbf{s}_0]_1 + c^2[\mathbf{s}_1]_1$		
	$\xrightarrow{[r']_1 = [r'_1]_1, [s']_1 = [s'_1]_1}$	
$\mathbf{a}' = c\mathbf{a}_0 + c^2\chi\mathbf{a}_1$ $\mathbf{b}' = c^{-1}\mathbf{b}_0 + c^{-2}\chi\mathbf{b}_1$		$[\mathbf{x}']_2 = ([x_1]_2, \ldots, [x_{\nu-1}]_2)$ $[\mathbf{y}']_2 = ([y_1]_2, \ldots, [y_{\nu-1}]_2)$ $[\alpha']_1 = c^{-1}[\alpha_{-1}]_1 + [\alpha]_1 + c[\alpha_1]_1$ $[\beta']_1 = c[\beta_{-1}]_1 + [\beta]_1 + c^{-1}[\beta_1]_1$ $z' = z_{-1}c + z + z_1c^{-1}$
		$e\left([r']_1 - c^{-1}[r]_1, [1]_2\right) \stackrel{?}{=} e\left(c^{-2}[r]_1, [x_\nu]_2\right)$ $e\left([s']_1 - c[s]_1, [1]_2\right) \stackrel{?}{=} e\left(c^2[s]_1, [y_\nu]_2\right)$
$\mathsf{w}' = ([\mathbf{r}']_1, [\mathbf{s}']_1, \mathbf{a}', \mathbf{b}')$		$\mathsf{x}' = ([r']_1, [s']_1, [\mathbf{x}']_2, [\mathbf{y}']_2, [\alpha']_1, [\beta']_1, z')$

Also, note that by construction of the commitment keys, for all $i \in \{1, \ldots, \nu\}$ and $j \in \{1, \ldots, 2^{i-1}\}$, it holds that $[r_{2^{i-1}+j}]_1 = x_i[r_j]_1$, which means that $[r']_1 = [r'_1]_1 = c^{-1}[r_1]_1 + c^{-2}[r_{2^{\nu-1}+1}]_1 = c^{-1}[r]_1 + c^{-2}x_\nu[r]_1$ and the verifier always accepts the pairing test.

It remains to show that $(\mathsf{par}, [r']_1, [\mathbf{r}']_1, [\mathbf{x}']_2) \in \mathcal{L}_{\mathsf{Com}}$. It is evident that $[r'_1]_1 = [r']_1$. We show that the various Diffie-Hellman Relations hold for the reduced statement.

Let $i \in \{1, \ldots, \nu - 1\}$ and $j \in \{1, \ldots, 2^{i-1}\}$. It holds that $[r'_{2^{i-1}+j}]_1 = x_i[r'_j]_1$. Indeed,

$$\begin{aligned}[r'_{2^{i-1}+j}]_1 &= c^{-1}[r_{2^{i-1}+j}]_1 + c^{-2}[r_{2^{\nu-1}+2^{i-1}+j}]_1 = c^{-1}x_i[r_j]_1 + x_\nu x_i c^{-2}[r_j]_1 \\ &= x_i(c^{-1}[r_j]_1 + x_\nu c^{-2}[r_j]_1) = x_i[r'_j]_1.\end{aligned}$$

Similar calculations show the part related to $\mathbf{s}'$. We can now argue completeness exactly as in the $\mathcal{U}_{2^\nu}$ case.

Witness extended emulation: For witness extended emulation we need to prove that, for each round, we can extract the witness, i.e. the commitment key and the commitment openings w.r.t. it. We show next how to extract the commitment keys. After having these, we can argue as in [BCC+16] except that we use the corresponding $\mathcal{X}_{\nu,1}$-FindRep assumption.

Assume we get two accepting transcripts for different challenges c from the prover. We show that given a witness for the reduced statement, we can extract the unique valid commitment keys $[\mathbf{r}]_1, [\mathbf{s}]_1$.

Let $[\mathbf{r}'_b]_1 = c_b^{-1}[\mathbf{r}_0]_1 + c_b^{-2}[\mathbf{r}_1]_1$ be the new commitment keys for two different challenges c_0, c_1. The matrix with rows (c_b^{-1}, c_b^{-2}) for $b \in \{0, 1\}$ is invertible, so we can take appropriate linear combination and extract $[\mathbf{r}_0]_1$, $[\mathbf{r}_1]_1$. We show that this is the commitment key. First note that since the transcript is accepting, we have that for both reduced keys $[r'_{2^{i-1}+j}]_1 = x_i[r'_j]_1$ which means that $[r_{2^{i-1}+j}]_1 = x_i[r_j]_1$ and $[r_{2^{\nu-1}+2^{i-1}+j}]_1 = x_i[r_{2^{\nu-1}+j}]_1$ for all $i \leq \nu - 1, j \leq 2^i$. In other words $[\mathbf{r}_0]_1$ and $[\mathbf{r}_1]_1$ are valid commitment keys w.r.t. the same $[x_1]_2, \ldots, [x_{\nu-1}]_2$. By the pairing test, we have that $[r'_b]_1 = c_b^{-1}[r]_1 + c_b^{-2}x_\nu[r]_1 = c_b^{-1}[r_{0,1}]_1 + c_b^{-2}[r_{1,1}]_1$. This equation holds for both challenges c_b, so it should be the case that $[r_{0,1}]_1 = [r]$ and $[r_{1,1}]_1 = x_\nu[r]$, thus the extracted key should be the unique key determined by $[x_1]_1, \ldots, [x_\nu]_1$. We argue for $[\mathbf{s}]_1$ in the same way. After extracting the keys the extractor works exactly as in [BCC+16] to extract $\mathbf{a}, \mathbf{b}$.

Updatability: The updatability property follows by noting that the reference string contains no values apart from the commitment key which is updatable as shown in Thm. 4.

Complexity: It is evident that the protocol needs ν rounds. In each round the size of the witness is decreased in half, and we perform a constant number of communication, so we

have $O(\nu)$ communication complexity. The prover in round i performs $O(2^{\nu+i-1})$ computations, so the prover complexity is $O\left(\sum_{i=1}^{\nu} 2^{\nu-i+1}\right) = O(2^{\nu})$, while the verifier does $O(1)$ operations and therefore its complexity is $O(\nu)$. □

3.3 Polynomial Commitment Scheme

The inner product argument presented in the previous section is a very powerfull primitive. Indeed, many relations of practical use can be expressed as an inner product relation; for example, one can derive vector and polynomial commitment schemes by simply expressing them as inner products. The efficiency and security properties of such construction are essentially the same as the inner product argument itself.

Directly using the inner product argument can also yield a universal non-interactive argument of knowledge for **NP** without the need of any additional assumption (such construcitons only require the security of the random oracle which is either way considered to hold for the inner product argument itself). For example, one could use the techniques introduced in Sonic [MBKM19] which build on the information theoretic part of the argument of [BCC+16]. Importantly, this transformation does not require any privacy property from the inner product arguments such as zero knowledge, hence we can directly derive such a construction using the protocol of Fig. 3.4.

The resulting protocol is a universal non-interactive argument for **NP** that relies only in the security of the random oracle and the FindRep assumption used to instantiate the inner product argument parameters. Specifically, it does not need any additional "non-falsifiable" assumptions such as knowledge-type assumptions or the algebraic group model.

While this is our ultimate goal, in this section we achieve it in a different way that fits better to the current abstractions of universal non-interactive arguments. We show how to instantiate a (hiding) polynomial commitment scheme using the inner product argument. Having such a primitive, we can rely on generic transformations to derive the argument for **NP** [CHM+20].

The biggest obstacle in doing so is that the construction of Fig. 3.4 is not zero knowledge; apart from the last round where we actually reveal the witness -which holds information about the initial witness- we also reveal information about the cross terms in each round. Nevertheless, there are known techniques to randomize the leaked infromation and derive hiding polynomial commitments (see for example [BGH19; BCMS20]). The goal of this section is to adapt these techniques to work with the improved inner product arguement, thus, yielding a hiding polynomial commitment scheme secure in random oracle model under one of the considered FindRep assumptions. This results to a univeral and updateable non-interactive zero knowledge argument for **NP** under the same assuptions.

3.3.1 Non-Hiding Polynomial Relation Argument

In this section we show how to modify the construction of Fig. 3.4 to function as a argument for the relation of polynomial commitment opening. Until the end of this section, we consider the isomorphism $p \in \mathbb{F}^{<2^\nu}[X] \Leftrightarrow \mathbf{p} \in \mathbb{F}^{2^\nu}$. We denote with boldface font the coefficients of the polynomial p. We commit to a polynomial p by simply committing to the vector $\mathbf{p}$ consisting of its coefficients, that is, we compute $[\alpha]_1 = [\mathbf{r}]_1^\top \mathbf{p}$.

Now, suppose we want to convince a verifier that $p(v) = z$ for some public values z. We follow the template of the folding technique. The prover gives "cross term" commitments to encoding the low and high degree terms of the polynomial p (denoted p_0, p_1) and the values $v_0 = p_0(z)$ and $v_1 = p_1(z)$. The latter are the evaluations of two polynomials with degree half of that of p on the point z. Then the prover and the verifier combine the commitments based on a challenge from the verifier and reduce the statement to one of smaller size.

Polynomial relation construction. We next describe the argument. We start by defining the language of the argument. We sample the commitment key from the multilinear distribution $\mathcal{X}_{\nu,1}$, but as we have discussed, one can use the univariate equivalent which is a special case. The language is defined as follows:

$$
\begin{aligned}
([r]_1, [\mathbf{x}]_2, [\alpha]_1, z, v) \in \mathcal{L}_{\mathsf{PC}} \iff \\
\exists\, [\mathbf{r}]_1 \in \mathbb{G}_1^{2^\nu}, \mathbf{p} \in \mathbb{F}^{2^\nu} \text{ s.t.} \\
([r]_1, [\mathbf{r}]_1, [\mathbf{x}]_2) \in \mathcal{L}_{\mathsf{Com}}^{\mathcal{X}_{\nu,1}} \wedge \\
[\alpha]_1 = [\mathbf{r}^\top \mathbf{p}]_1 \wedge p(v) = z.
\end{aligned}
$$

We present the recursive step of the protocol in Fig. 3.5. As in the previous case, when the statement is small the prover simply reveals the witness.

We next show that the above protocol is indeed an argument for the language $\mathcal{L}_{\mathsf{PC}}$.

Theorem 6. *Protocol of Fig. 3.5 is a public coin, argument of knowledge for the relation $\mathcal{L}_{\mathsf{PC}}$ with ν round complexity, $O(2^\nu)$ prover complexity, and $O(\nu)$ communication and verification complexity under either the $\mathcal{X}_{\nu,1}$-FindRep assumption (or the $\mathcal{X}_{2^\nu}$-FindRep assumption). The argument yields an updateable non-interactive argument of knowledge in the random oracle model. In the former case the proof size of an update is $O(\nu)$ and in the latter $O(1)$.*

Proof.
Completeness: We show that each reduction round leads to a valid reduced statement. The part of the statement regarding the key is dealt in the exact same way as the inner product argument case so we omit it here. We next show that the resulting statement-witness pair is a valid one for $\mathcal{L}_{\mathsf{PC}}$.

Figure 3.5 Recursive step of the polynomial relation argument. When the statement size is small (constant), the prover sends the witness to the verifier.

$\mathsf{x} = ([r]_1, [\mathbf{x}]_2, [g]_1, [\alpha]_1, v, z)$, $\mathsf{w} = ([\mathbf{r}]_1, \mathbf{p})$,
claim: $\mathsf{x} \in \mathcal{L}_{\mathsf{PC}}$

$\mathcal{P} : \mathsf{x}, \mathsf{w}$		$\mathcal{V} : \mathsf{x}$
$v_0 = p_0(z)$		
$[\alpha_{-1}]_1 = [\mathbf{r}_1]_1^\top \mathbf{p}_0$		
$[\alpha_1]_1 = [\mathbf{r}_0]_1^\top \mathbf{p}_1$		
	$\xrightarrow{[\alpha_{-1}]_1, [\alpha_1]_1, v_0}$	
	$\xleftarrow{c}$	$c \leftarrow \mathbb{F}$
$[\mathbf{r}']_1 = [\mathbf{r}_0]_1 + c^{-1}[\mathbf{r}_1]_1$		
	$\xrightarrow{[r']_1 = [r'_1]_1}$	
$\mathbf{p}' = \mathbf{p}_0 + c\mathbf{p}_1$		$[\mathbf{x}']_2 = ([x_1]_2, \ldots, [x_{\nu-1}]_2)$
		$[\alpha']_1 = c^{-1}[\alpha_{-1}]_1 + [\alpha]_1 + c[\alpha_1]_1$
		$v_1 = (v - v_0)z^{1-\nu}$
		$v' = v_0 + cv_1$
		$e\left([r']_1 - c^{-1}[r]_1, [1]_2\right) =$ $e\left(c^{-2}[r]_1, [x_\nu]_2\right)$
$\mathsf{w}' = (\mathbf{r}', \mathbf{p}')$		$\mathsf{x}' = ([r']_1, [\mathbf{x}']_2, [\alpha']_1, z, v')$

First, we show that $\mathbf{p}' = \mathbf{p}_0 + \mathbf{p}_1$ is an opening of $[\alpha']_1$. We have

$$\begin{aligned}
[\alpha']_1 &= c^{-1}[\alpha_{-1}]_1 + [\alpha]_1 + c[\alpha_1]_1 \\
&= c^{-1}[\mathbf{r}_1]_1^\top \mathbf{p}_0 + [\mathbf{r}_0]_1^\top \mathbf{p}_0 + [\mathbf{r}_1]_1^\top \mathbf{p}_1 [\alpha]_1 + c[\mathbf{r}_0]_1^\top \mathbf{p}_1 \\
&= [\mathbf{r}_0]_1^\top (\mathbf{p}_0 + c\mathbf{p}_1) + [\mathbf{r}_1]_1^\top (c^{-1}\mathbf{p}_0 + \mathbf{p}_1) \\
&= [\mathbf{r}_0]_1^\top (\mathbf{p}_0 + c\mathbf{p}_1) + c^{-1}[\mathbf{r}_1]_1^\top (\mathbf{p}_0 + c\mathbf{p}_1) \\
&= ([\mathbf{r}_0]_1 + c^{-1}[\mathbf{r}_1]_1)^\top (\mathbf{p}_0 + c\mathbf{p}_1) = [\mathbf{r}']_1^\top \mathbf{p}'
\end{aligned}$$

It remains to show that the polynomial p' is evaluated to v' at z. Note that this corresponds to the evaluation $p_0(z) + cp_1(z)$, so it is enough to show that this is the value of v'. Since $v' = v_0 + cv_1$ and $v_0 = p_0(z)$, it is enough to assert that $v_1 = p_1(z)$. In what follows, note that we can write the evaluation of p at z as $v_0 + z^{n/2}v_1 = v_0 + z^{\nu-1}v_1$. We have

$$v_1 = (v - v_0)z^{1-\nu} \Leftrightarrow v_1 z^{\nu-1} = v - v_0 \Leftrightarrow v_0 + v_1 z^{\nu-1} = v$$

and the evaluation of p' at z is indeed v'.

Witness extended emulation: For witness extended emulation we need to prove that, for each round, we can extract the witness, i.e. the commitment key and the commitment openings w.r.t. it. The part of the statement regarding the key is dealt in the exact same way as the inner product argument case so we omit it here. We next show how to extract a polynomial p such that $p(z) = v$, given three statement/witness pair of the next round x', w'.

Consider three such statement/witness pairs

$$\mathsf{x}_i = ([r'_i]_1, [\mathbf{x}'_i]_2, [\alpha'_i]_1, z, v_i), \qquad \mathsf{w}_i = (\mathbf{r}'_i, \mathbf{p}'_i)$$

for $i \in \{1, 2, 3\}$ and let $\mathbf{r}'_i$ be a valid commitment key for each. Consider the relations that are satisfied for each in the field:

$$\begin{pmatrix} \alpha'_1 \\ \alpha'_2 \\ \alpha'_3 \end{pmatrix} = \begin{pmatrix} \mathbf{r}'^{\top}_1 \mathbf{p}'_1 \\ \mathbf{r}'^{\top}_2 \mathbf{p}'_2 \\ \mathbf{r}'^{\top}_3 \mathbf{p}'_3 \end{pmatrix} = \begin{pmatrix} (\mathbf{r}_0 \ \ \mathbf{r}_1)(\mathbf{p}'_1 \ \ c^{-1}\mathbf{p}'_1) \\ (\mathbf{r}_0 \ \ \mathbf{r}_1)(\mathbf{p}'_2 \ \ c^{-1}\mathbf{p}'_1) \\ (\mathbf{r}_0 \ \ \mathbf{r}_1)(\mathbf{p}'_3 \ \ c^{-1}\mathbf{p}'_1) \end{pmatrix} = \begin{pmatrix} \mathbf{r}(\mathbf{p}'_1 \ \ c_1^{-1}\mathbf{p}'_1) \\ \mathbf{r}(\mathbf{p}'_2 \ \ c_2^{-1}\mathbf{p}'_1) \\ \mathbf{r}(\mathbf{p}'_3 \ \ c_3^{-1}\mathbf{p}'_1) \end{pmatrix} = \begin{pmatrix} c_1^{-1} & 1 & c_1 \\ c_2^{-1} & 1 & c_2 \\ c_3^{-1} & 1 & c_3 \end{pmatrix} \begin{pmatrix} \alpha_{-1} \\ \alpha \\ \alpha_1 \end{pmatrix}$$

and note that the matrix with the challenges c_1, c_2, c_3 is a shifted Vandemonde matrix, and hence invertible. Thus, there exists an efficiently computable vector $\mathbf{d} = (d_1, d_2, d_3)$ such that

$$\alpha = \mathbf{d}^{\top} \begin{pmatrix} \mathbf{r}(\mathbf{p}'_1 \ \ c_1^{-1}\mathbf{p}'_1) \\ \mathbf{r}(\mathbf{p}'_2 \ \ c_2^{-1}\mathbf{p}'_1) \\ \mathbf{r}(\mathbf{p}'_3 \ \ c_3^{-1}\mathbf{p}'_1) \end{pmatrix} = \mathbf{d}^{\top} \begin{pmatrix} \mathbf{r}\tilde{\mathbf{p}}_1 \\ \mathbf{r}\tilde{\mathbf{p}}_2 \\ \mathbf{r}\tilde{\mathbf{p}}_2 \end{pmatrix}$$

and we can extract $\mathbf{p}$ by setting $\mathbf{p} = d_1\tilde{\mathbf{p}}_1 + d_2\tilde{\mathbf{p}}_2 + d_3\tilde{\mathbf{p}}_3$. Finally, we show that the extracted polynomial evaluated at z yeilds v. Note that in the same way we can extract valid openings for the commitments α_{-1} and α_1. We claim that these commitments are w.r.t. keys $\mathbf{r}_1, \mathbf{r}_0$ respectively. Indeed, assume this is not the case and consider an honest execution with the extracted commitment $\mathbf{p}$ resulting in $\tilde{\alpha}_{-1}, \tilde{\alpha}_1$ cross term commitments. Then, viewing the challenge as a formal variable X

$$X^{-1}[\alpha_{-1}]_1 + [\alpha]_1 + X[\alpha_1]_1 = X^{-1}[\tilde{\alpha}_{-1}]_1 + [\alpha]_1 + X[\tilde{\alpha}_1]_1$$

we have that w.o.p. over the choice of the challenge c the honest and claimed cross term commitments are equal. Having two different openings implies we we break the FindRep assumption for the key $\mathbf{r}$.

To assert that the opening of the extracted polynomial is indeed v' in each case, it is enough to assert that the openings of the cross-term polynomials are v_0 and v_1. We claim that w.o.p. the extracted openings $\mathbf{p}_0$ and $\mathbf{p}_1$ of the commitments α_{-1}, α_1 correspond to polynomials that evaluate at v_0 and v_1 respectively at z. It is enough to note that the values v_0, v_1 are claimed before the challenge c is set. If they are not valid openings for p_0, p_1, then, except with negligible probability, the polynomial $p' = p_0 + cp_1$ will not have a valid opening $v' = v_0 + cv_1$ at z, contradicting the success of the adversary.

Updateability and complexity are the same as in the proof of Thm. 5. □

Augmenting with low degree test. The polynomial commitment presented does not guarantee that the degree of the polynomial considered is of bounded degree d. It only guarantees that its degree is less than n where n is the size of the commitment key. We next discuss how to overcome this limitation. The techniques is quite simple. The prover claims that p is a polynomial of degree less than $d < n$. To convince the verifier for this fact, it simply commits to the polynomial $p'(X) = X^{n-d}p(X)$ and shows consistency of the two polynoimals as follows:

- The verifier sends a random challenge z.
- The prover gives the values $v' = p'(z)$ and $v = p(z)$.
- The verifier asserts that $v' = z^{n-d}v$.
- The prover and verifier execute in parallel two protocols for the two claims as described in Fig. 3.5.

This is enough since either the polynomial relation $p'(X) = X^{n-d}p(X)$ holds, or the prover will be caught lying with overwhelming probability due to the Schwartz–Zippel lemma. Note that the prover and verifier can reduce the two claims to a single claim by considering the polynomial $p'(X) + \alpha p(X)$ based on a challenge of the verifier.

Supporting hiding commitments. The final addition we need is the support for hiding commitments. This is crucial in the case of zero knowledge since normally the polynomial encodes the statement witness we want to hide. The modifications we need are (1) give an extra element $[\rho]_1$ for the commitment key, which will be used as a blinding factor and (2) before giving a proof of an evaluation, we blind the polynomial.

A bit more concretely, the prover commits to $p(X)$ by giving to the verifier the value $[\alpha]_1 = [\mathbf{r}]_1^\top \mathbf{p} + [\rho]_1 \tau$ where τ is sampled uniformly from the field. Note that this makes the element $[\alpha]_1$ uniformly distributed over $\mathbb{G}_1$ and thus reveals no information about $p(X)$. Finally, when the prover claims that $p(v) = z$, it also gives a commitment to a uniformly distributed polynomial $p'(X)$ conditioned on $p'(X) = 0$. The verifier responds with a challenge c and the prover and verifier proceed in interacting for proving the (non-hiding) claim $(p + cp')(v) = z$. Note that the polynomial p is now blinded and therefore no information is leaked about it.

The above procedure can be adapted easily for also checking a degree bound on d.

3.3.2 Polynomial Commitment Construction

In this section, we describe the whole polynomial commitment construction. We describe the opening algorithm as an interactive public coin protocol between the prover and the verifier.

Essentially, the construction is the interactive protocol we presented, where we adapt it to support degree checking and hiding. The construction is presented in Fig. 3.6. As we did for the previous constructions, we only consider the case of the $\mathcal{X}_{\nu,1}$ distribution.

Theorem 7. *The polynomial commitment derived by applying the Fiat-Shamir transform in Protocol of Fig. 3.6 is a hiding polynomial commitment that is extractable under the* $\mathcal{X}_{\nu,1}$-FindRep *assumption assuming the security of the Fiat-Shamir transform.*

Proof. Correctness follows by inspection of the protocol and completeness of the interactive protocol of Fig. 3.5. We next show extractability. By knowledge soundness of the protocol of Fig. 3.5, we can extract valid witnesses for each of the three claims. That is, we can extract polynomials $q(X), q'(X)$ that are valid commitments to $[\beta]_1, [\beta']_1$ respectively and satisfy:

$$q(z) = v, \qquad q(\omega) = v_q, \qquad q'(\omega) = v_{q'}$$

First, we claim that the polynomial q is of degree less than d with overwhelming probability. Indeed, note that both polynomials q, q' are of degree at most D and that $q'(\omega) = \omega^{D-d} q(\omega)$ holds for a random point that is fixed after the polynomials are set. Thus, with overwhelming probability, this relation holds as a polynomial identity, in which case the degree of ω is bounded by D.

Next, we show that we can extract valid witnesses for the commitments $[\alpha]_1, [\alpha']_1$. Assume we have two valid executions with for two different challenges c_1, c_2. Then we have for $i \in \{1, 2\}$:

$$[\beta_i]_1 + [\rho]_1 \tau_i'' = [\alpha]_1 + c_i [\alpha']_1$$

and since we have openings for α, α', we can get openings for the commitments $[\alpha]_1, [\alpha']_1$. Furthermore, these openings corresponds to polynomials of degree less than d unless we break the binding of the commitment scheme: the coefficients corresponding to higher degree terms will be zero for the commitments $[\beta_i]_1$, so if this is not the case, we get two different openings for the same commitments. Note that this also implies that $q(X) = p(X) + cp'(X)$, otherwise we break the binding property of the Pedersen commitment scheme.

Finally, we need to show that the evaluation of the extracted commitment $\mathbf{p}$ corresponding to $[\alpha]_1$ is indeed v at z. Now, with overwhelming probability over the choice of c, we have that $q(z) = p(z) + cp'(z)$. For this relaiton to hold for all c, it must be the case that $q(z) = p(z) = v$ and $p'(z) = 0$ which concludes the claim.

Finally, we show that the opening protocol is honest verifier zero knowledge. Note that this is enough for proving hiding since both the commitment itself reveals no information for $p(X)$ due to the blinding factor τ and the evaluation itselfs leaks nothing apart from the claim $p(z) = v$. We show how to simulate the execution transcript.

First, we sample $\omega \leftarrow \mathbb{F}$ and uniform polynomials q' and q conditioned on $\deg_q < d$, $q(\omega) = v_q, q'(\omega) = v_{q'}$ and $q(z) = v$. We compute honestly (non-hiding) commitments

Figure 3.6 A polynomial commitment based on the folding technique. We describe the opening/verification algorithms as a public coin interactive protocol between a prover and a verifier. A pair of algorithms can be derived from the protocol using the FS transform.

$\mathsf{KeyGen}(1^\lambda, D = 2^\nu)$**:**

$\mathsf{gk} \leftarrow \mathcal{G}(1^\kappa)$

$\mathbf{r} \leftarrow \mathcal{X}_{1,\nu}$, $\rho \leftarrow \mathbb{F}$

Let $\mathbf{r} = \overline{\mathbf{x}}$ for $\mathbf{x} = (x_\nu, \ldots, x_1)$

Output $\mathsf{ck} = ([\mathbf{r}]_1, [\rho]_1, [\mathbf{x}]_2)$

$\mathsf{Com}(\mathsf{ck}, p(X))$**:**

Let $\mathbf{p} \in \mathbb{F}^{2^\nu}$ be the coefficients of $p(X)$

Sample $\tau \leftarrow \mathbb{F}$

Output $[\alpha]_1 = [\mathbf{r}]_1^\top \mathbf{p} + [\rho]_1 \tau$ and $\mathsf{aux} = \tau$.

Opening protocol:

$\mathsf{x} = (\mathsf{ck}, \alpha, d, z, v)$, $\mathsf{w} = (\mathbf{p}, \tau)$

Claim: $[\alpha]_1 = [\mathbf{r}]_1^\top \mathbf{p} + [\rho]_1 \tau$, $\deg_p < d$ and $p(z) = v$

– $\mathcal{P}$:

  Sample $p'(X) \in \mathbb{F}[X]$ uniformly conditioned on $\deg_{p'} < d$ and $p'(z) = 0$

  Sample $\tau' \leftarrow \mathbb{F}$.

  Send $[\alpha']_1 = [\mathbf{r}]_1^\top \mathbf{p}' + [\rho]_1 \tau'$ to $\mathcal{V}$

– $\mathcal{V}$: Sample $c \leftarrow \mathbb{F}$ and send it to $\mathcal{P}$

– $\mathcal{P}$:

  Set $q(X) = p(X) + cp'(X)$

  Set $\tau'' = \tau + c\tau'$

  Set $q'(X) = X^{D-d} q(X)$ and $[\beta']_1 = [\mathbf{r}]_1^\top \mathbf{q}'$

  Send τ'', $[\beta']_1$ to $\mathcal{V}$

– $\mathcal{V}$: Sample $\omega \leftarrow \mathbb{F}$ and send it to $\mathcal{P}$

– $\mathcal{P}$ sends $v_q = q(\omega)$, $v_{q'} = q'(\omega)$

– $\mathcal{V}$ rejects if $v_{q'} \neq \omega^{D-d} v_q$.

– $\mathcal{P}$ and $\mathcal{V}$ compute commitment $[\beta]_1 = [\alpha]_1 + c[\alpha']_1 - [\rho]_1 \tau''$.

– $\mathcal{P}$ and $\mathcal{V}$ proceed as in Fig. 3.5 in proofs of the claims:

  1. $[r]_1, [\mathbf{x}]_2, [\beta]_1, z, v) \in \mathcal{L}_{\mathsf{PC}}$
  2. $[r]_1, [\mathbf{x}]_2, [\beta]_1, \omega, v_q) \in \mathcal{L}_{\mathsf{PC}}$
  3. $[r]_1, [\mathbf{x}]_2, [\beta']_1, \omega, v_{q'}) \in \mathcal{L}_{\mathsf{PC}}$

$[\beta]_1, [\beta']_1$ for these values and proceed to an honest execution of the three instances of the protocol of Fig. 3.5. We then sample $c, \tau'' \leftarrow \mathbb{F}$ and set $[\alpha']_1 = c^{-1}([\beta]_1 - [\alpha]_1 + [\rho]_1\tau'')$. It is easy to assert that the produced transcript is identically distributed to an honest one: the values $[\alpha']_1, c, \tau'''$ are uniformly distributed conditioned on $[\beta]_1 = [\alpha]_1 + c[\alpha']_1 - [\rho]_1\tau''$ and the rest of the values are computed honestly. □

Chapter 4

Fully-succinct, Publicly Verifiable Delegation from Constant-Size Assumptions

This chapter is based on the paper "Fully-succinct, Publicly Verifiable Delegation from Constant-Size Assumptions" that was presented in TCC 2021, which is joint work with Alonso González.

In a delegation scheme, a verifier with limited computational resources (a mobile device for example) wishes to delegate a heavy but still polynomial computation to an untrusted prover. The prover, with more computational power but still of polynomial time, computes a proof which the verifier accepts or rejects. Given the limitations of the verifier, the proof should be as short as possible and the verification process should consume as few computational resources as possible. Additionally, the construction of the proof should not be much costlier than performing the computation itself.

A delegation scheme can be easily constructed from a Succinct Non-Interactive Argument of Knowledge (SNARK) for **NP**. Schemes like [GGPR13; Gro16] are very appealing in practice because a proof consists of only a constant number of group elements and verification requires the evaluation of a constant number of pairings. The downside is that these SNARKs are based on strong and controversial assumptions such as the knowledge of exponent assumption or on idealized models such as the random oracle [BR93], the algebraic group model [FKL18] or the generic group model [Sho97].

Such assumptions are called non-falsifiable because there is no way of efficiently deciding whether an adversary breaks the assumption or not. In such assumptions, the adversary is

treated in a non black box way and the assumption argues about *how* an adversary performs a computation instead of *what* computation it cannot perform. Since SNARKs can handle even **NP** computations, soundness becomes an essentially non-falsifiable property where one needs to decide whether an adversary produces a true or false statement without any witness but only with a very short proof. Gentry and Wichs [GW11] proved that SNARKs for **NP** are (in a broad sense) impossible to construct without resorting to non-falsifiable assumptions.

While this impossibility result justifies the use of such assumptions for non-deterministic computation, this is not the case for delegation of computation which only considers deterministic computation. Indeed, in this case, soundness becomes an efficiently falsifiable statement: determining whether the adversary breaks soundness simply requires to evaluate the delegated polynomial computation on some input x and check whether it is accepting or rejecting. Actually, getting delegation from falsifiable assumptions is easy in general: let Π be a SNARK for **NP**. For a binary relation $\mathcal{R}$, the assumption "Π is sound for $\mathcal{R}$" is in general non-falsifiable since checking membership in the corresponding language is hard (assuming $\mathbf{P} \neq \mathbf{NP}$) and the SNARK proof does not help as shown by [GW11]. On the contrary, for a relation $\mathcal{R}$ in **P**, the assumption becomes falsifiable since one can efficiently compute $\mathcal{R}(x)$. Nevertheless, the important issue is to consider the *quality* of the assumption in place since the assumption "the proof system is sound" is tautological. Ideally, we should rely on simple and well understood assumptions *without* sacrificing other desirable properties.

Almost all known constructions that base their soundness on falsifiable assumptions (or even no assumptions at all) come with some compromises: they (1) are not expressive enough to capture all polynomial time computation [KPY18; GR19; CCH+19; JKKZ20] (2) are interactive [GKR08; RRR16], (3) are designated verifier [KRR13; KRR14; KP16; BHK17; BKK+18] or (4) rely on strong (yet falsifiable) assumptions related to obfuscation [CHJV15; KLW15; BGL+15; ACC+16; CCC+16] or multi-linear maps [PR17].

An exception to this is a construction of Kalai et al. [KPY19] of a delegation scheme for any poly-time computation based on a newly introduced q-size assumption in bilinear groups. The size of the assumption is $q = \log T$ and T is the time needed to perform the computation. As for efficiency, the size of the proof is polylog(T) group elements which becomes poly(κ) if $T \leq 2^{\kappa}$.

However, in spite of the recent progress, there's still a gap in the proof size and verification with respect to the most efficient known constructions, namely those based on paring based SNARKs.

Results

In this chapter we consider the question *"what are the simplest assumptions that imply publicly verifiable, non-interactive delegation of computation"?* Here "*simple*" should be interpreted as falsifiable and well understood. Having practicality in mind as well, we would also want a

delegation scheme that competes in efficiency with the most efficient constructions to date, namely those that are based on non-falsifiable assumptions.

The main goal of this chapter is the presentation of a fully-succinct, non-interactive, publicly verifiable delegation scheme from any k-Matrix Diffie-Hellman (k-MDDH) assumption for $k \geq 2$, as for example the decisional linear assumption (DLin) [BBS04]. In the more efficient setting of asymmetric groups, soundness can be based on the natural translation of DLin where the challenge is encoded in both groups (the SDLin assumption of [GHR15]). Here, by fully-succinct we mean that the proof size is linear in the security parameter and verification requires a linear number of operations (whose complexity depends only on the security parameter) in the size of the input of the computation. We achieve these goals but with the drawback that the prover computation and the size of the srs are quadratic in the size of the circuit. We summarize the main result in the next (informal) theorem.

Theorem 8. (Informal). *There exists a non-interactive, publicly verifiable delegation scheme for any polynomial size circuit C with n-size input that is adaptively sound under any k-MDDH assumption for $k \geq 2$ with the following efficiency properties: the* srs *size is* $\mathsf{poly}(\kappa)\,|C|^2$*, prover complexity is* $\mathsf{poly}(\kappa)\,|C|^2$*, proof size is* $\mathsf{poly}(\kappa)$ *and verification complexity is* $\mathsf{poly}(\kappa)n$.

Our construction is also concretely efficient as far as proof size and verification complexity are concerned. The proof comprises of $10+8$ group elements of an asymmetric bilinear group and verification requires n exponentiations plus 36 evaluations of the pairing function, where n is the size of the input. The attractive concrete efficiency is achieved due to the structure-preserving nature [AFG+16] of our construction. This notion captures that all algorithms solely perform group operations, namely they are *algebraic*, and there is no need to encode cryptographic primitives such as hash functions or pairings as arithmetic circuits, a process that is very inefficient in practice.

This result demonstrates two things. First, delegation of computation can be based on very simple, standard assumptions. Second, its structure preserving nature hints to the plausibility of practically efficient delegation schemes comparable in efficiency with the ones based on SNARKs, but under simple, standard assumptions. In Table 4.1 we present a comparison of our delegation of computation construction with other pairing based schemes.

No-Signaling SSB Commitments and Succinct Pairing-based Quasi-Arguments. We follow and extend the ideas of Paneth and Rothblum [PR17] and Kalai et al. [KPY19] for constructing delegation schemes for poly-time computations from what they called quasi-arguments of knowledge with no-signaling extractors. First, we formalize a similar notion for commitment schemes and show that the somewhere statistically binding (SSB) commitments of [GHR15; FLPS20] are no-signaling when they also have what we call an "oblivious trapdoor generator". Second, we use the no-signaling SSB commitments to construct more efficient constant-sized quasi-arguments of knowledge for linear and quadratic relations. We achieve this by combining SSB commitments with the very efficient quasi-adaptive non-interactive zero-knowledge arguments for linear [JR13; LPJY13; JR14; KW15] and quadratic relations [GHR15;

	Language	Verification	Proof size	srs size	Assumption
[GGPR13; Gro16]	AC	$n\mathsf{e} + \mathcal{O}(1)\mathsf{p}$	$\mathcal{O}(\kappa)$	$\mathcal{O}(\lvert C\rvert\,\kappa)$	Non Falsifiable
[KPY19] (base case)	RM	$n\mathsf{e} + \mathrm{poly}(\log d)\mathsf{p}$	$\mathcal{O}(\kappa \log d)$	$O((n+d)\kappa)$	$\log d$-Assumption
[GR19]	AC	$n\mathsf{e} + \mathcal{O}(d)\mathsf{p}$	$\mathcal{O}(d\kappa)$	$\mathcal{O}(\lvert C\rvert\kappa)$	s-Assumption
This work	AC	$n\mathsf{e} + \mathcal{O}(1)\mathsf{p}$	$\mathcal{O}(\kappa)$	$O(\lvert C\rvert^2\,\kappa)$	DLin/SDLin

Table 4.1: Comparison between different pairing based delegation schemes and our results. Verification is given in number exponentiations (e) and pairings (p). d is the circuit depth/number of steps of a computation, n the number of inputs, s the circuit width/computation space and $|C|$ the circuit size. AC stands for "Arithmetic Circuit" and RM for "RAM Machine". For [KPY19] we only consider the "base case" and not the "bootstrapped" constructions, because bootstrapping adds a considerable overhead and is thus incomparable in terms of group operations. We stress out, however, that the srs size of the bootstrapped construction is sublinear in the time of the computation.

	Language	Verification	Proof size	CRS size	Assumption
[GOS06]	AC	$\mathcal{O}(\lvert C\rvert)\mathsf{p}$	$\mathcal{O}(\lvert C\rvert\,\kappa)$	$\mathcal{O}(\kappa)$	SXDH
[GGPR13; Gro16]	AC	$\mathcal{O}(1)\mathsf{p}$	$\mathcal{O}(\kappa)$	$\mathcal{O}(\lvert C\rvert\,\kappa)$	Non Falsifiable
[GR19]	BC	$\mathcal{O}(n+d)\mathsf{p}$	$\mathcal{O}((n+d)\kappa)$	$\mathcal{O}(\lvert C\rvert\,\kappa)$	s-Assumption
[KNYY20]	NC^1	$\mathcal{O}(\lvert C\rvert)\mathrm{poly}(\kappa)$	$n\mathrm{poly}(\kappa)$	$\mathrm{poly}(\lvert C\rvert, \kappa, 2^d)$	DLin
This work	BC	$\mathcal{O}(n)\mathsf{p}$	$nO(\kappa)$	$\mathcal{O}(\lvert C\rvert^2\,\kappa)$	DLin/SDLin

Table 4.2: Comparison between different pairing based NIZK schemes and our results. Verification is given in number of pairings p. d is the circuit depth, n the number of (public and secret) inputs, s the circuit width and $|C|$ the circuit size. AC stands for "Arithmetic Circuit", BC for "Boolean Circuit" and NC^1 for constant depth boolean circuits.

DGP+19]. To this aim, we also show that the QA-NIZK arguments can be easily modified to have no-signaling extractors under standard assumptions.

Applications to NIZK. Our construction can be turned into a NIZK argument for **NP** of size $n + \mathcal{O}(1)$ group elements -namely $\mathcal{O}(n\kappa)$ proof size- under the same assumptions where n is the number of public and secret inputs of the circuit. In table 4.2 we provide a comparison of our NIZK construction and the literature. Using standard techniques, the argument implies compact NIZK for **NP** with proof size $\mathcal{O}(n) + \mathrm{poly}(\kappa)$. That is, the size of the proof is proportional to the size of the input and the security parameter only gives an additive overhead. In comparison, the state of the art is $O(|C|) + \mathrm{poly}(\kappa)$ for poly-sized boolean circuits and $\mathcal{O}(n) + \mathrm{poly}(\kappa)$ for log-depth boolean circuits [KNYY19; KNYY20]. We note that a similar result can be obtained by [KPY19], albeit with a stronger assumption.

Our argument can be also used to construct zk-SNARKs from quantitatively weaker assumptions than the state of the art. Indeed, the strongest assumption used in zk-SNARKs such as [GGPR13; Gro16] is a knowledge assumption which states that an adversary computing some elements of a bilinear group, satisfying a particular relation, must know their discrete logarithms.[1] Such assumption is used to extract an assignment to each of the circuit wires.

[1]Actually, the adversary must know a representation of these values as a linear combination of a set of

The "size" of such assumption is proportional to the number of extracted values, which in this case is the size of the circuit. Since our argument only requires the reduction to know the input of the circuit, we can rely on a knowledge assumption only for extracting the input. As a consequence the size of the assumption is drastically shortened. Since these assumptions are stronger as the size of the assumption increases and given that we lack good understanding of them, it is always safer to rely on shorter assumptions. Also, weaker assumptions translates to better concrete efficiency by using smaller security parameters.[2]

4.1 Technical Overview

To construct the delegation scheme we follow a commit-and-prove approach, which means that we first commit to the witness (the satisfying assignment of wires in a circuit) and then show that this witness satisfies some relation. We use somewhere statistically binding (SSB) commitments as those used in [GHR15; GR16; FLPS20] and show that they satisfy a *no-signaling extraction* property. Then, we do the same for the so called quasi-adaptive NIZK arguments for linear spaces [JR13; LPJY13; JR14; KW15] and for quadratic relations [GHR15; DGP+19]. From these primitives we can construct delegation for bounded-space computations/bounded width circuits with proof-size independent of the depth of the computation by following the techniques of [PR17; KPY19]. To get a succinct proof-size, in addition to the "depth compression", we must also perform a "width compression". To this end, we use ideas from the delegation scheme for bounded depth computations of González and Ráfols [GR19] and remove the necessity of a q-assumption to rely solely on constant size assumptions. To combine both "compressions" efficiently we exploit the fact that [GR19] is structure preserving and the verifier is a bounded width circuit. In the next sections we present these techniques.

4.1.1 No-Signaling Somewhere Statistically Binding Commitments

Somewhere statistically binding (SSB) hashing/commitments[3] were introduced by Hubacek and Wichs [HW15] and then improved by [OPWW15], and have been used for constructing efficient NIZK proofs [GHR15; GR16] as well as ring signatures [BDH+19].

An SSB commitment scheme is a generalization of dual mode commitments [GS08] where the commitment key can be sampled from many computationally indistinguishable distributions, each of which is making the commitments statistically binding for a number of K coordinates of the committed value. That is, when committing to a vector $\mathbf{m} = (m_1, \ldots, m_n)$ with a

group elements that she receives as input.

[2]We note, however, that in the case of non-falsifiable assumptions it not clear how an appropriate security parameter should be chosen.

[3]Through this paper we will refer to "commitments" while technically they are "hashes". We do so because in the context of NIZK proofs, it is traditional to commit to the witness and then prove that the committed value satisfy some relation. However, since we are less interested in zero-knowledge, the randomness of such commitments is 0 (or fixed/inexistent) and we end up with hashes.

commitment key ck_S associated with a set $S \subseteq [n]$ of size at most K, no (even computationally unbounded) adversary can compute a commitment c and two valid openings $\mathbf{m}, \mathbf{m}'$ such that for some $i \in S$ it holds that $\mathsf{m}_i \neq \mathsf{m}'_i$, except with negligible probability. Importantly, the size of the commitment c should be independent of n but may depend on the value K.

Known SSB commitments constructions are also extractable[4], that is, there exists an efficient algorithm that has some trapdoor information associated with ck_S and can efficiently extract from a commitment c a valid opening $(\mathsf{m}_i)_{i \in S}$. Note that the notion of a "valid opening" is well-defined due to the statistical binding property on the set S.

We argue that the SSB extractor has many similarities with the no-signaling extractors of [PR17; KPY19]. First, we briefly recall what a no-signaling extractor is in the context of quasi arguments of knowledge. A quasi argument is a proof system for a relation that defines some local constraints on the statement/witness pair. The requirement is that there exists a *no signaling extractor* that allows extracting a part of the witness from a verifying proof that is locally correct. Furthermore, each part of the extracted local witness can be in a sense extracted independently. This is formalized by requiring that extracting local witness w_S for a set S and restricting it to the variables $S' \subseteq S$ is computationally indistinguishable from extracting $\mathsf{w}_{S'}$ for the set S'. As we shall see shortly, this property is extremely useful when constructing delegation schemes.

In the case of SSB commitments, extractability of the local opening is just a local soundness guarantee. Additionally, indistinguishability of the commitment keys is a weaker form of the no-signaling property. Indeed, a no-signaling extractor must produce commitment keys which are indistinguishable for the various possible extractable sets. Otherwise a distinguisher for sets S, S' can be used for wining in the no-signaling game even without the extracted value. Nevertheless, this alone does not satisfy the no-signaling property: some information about the positions where the srs is programmed to extract might be revealed by (parts of) the extracted local openings.

We strengthen the indistinguishability property of the distributions of the commitment keys of SSB commitments to give them a no-signaling flavor. Roughly speaking, we require that the distributions of the commitment keys are computationally indistinguishable *even if the adversary has access to local openings associated with a set S' of committed values*. These local openings trivially reveal information about the set S' but we require that they do not leak information about the values outside of S'. That is, for any sets $S' \subseteq S$ of size at most K, the commitment keys $\mathsf{ck}_S, \mathsf{ck}_{S'}$ are computationally indistinguishable even if we allow the distinguisher access to local openings of S'.

Remark (Connection with PIR)**.** Somewhere statistically binding commitments/hashing is closely related with single server Private Information Retrieval Schemes (PIR) when the

[4]In the context of bilinear groups, we can consider f-extraction where one only extracts f applied to the witness. In particular, it is usual to consider f the (one-way) function that maps elements in $\mathbb{F}$ to one of the base groups $\mathbb{G}_1$ or $\mathbb{G}_2$. This is the notion of extractability we use in this work and is enough to obtain our results.

SSB commitment is also extractable. Indeed, we can think of the commitment key for an index i of the SSB as a PIR query and the commitment/hash as the PIR answer. Then, one can decode the PIR query using the trapdoor associated with the commitment key. The SSB commitments we use are different from PIRs in three ways: (1) we do not extract the PIR answers, but we f-extract, specifically we extract encodings of messages in a group but not their discrete logarithms, (2) we directly use SSBs with locality greater than one instead of making parallel PIR queries to improve concrete efficiency and (3) the size of the commitment key is proportional to the size of the committed values, while in PIRs the query should be small compared to the database size. Furthermore, we exploit in a non-black box way the properties as well as the algebraic structure of the SSB commitments to compose them with other protocols, such as group based quasi-adaptive non-interactive zero knowledge arguments.

4.1.1.1 SSB Commitments with Oblivious Trapdoor Generation.

We define a stronger notion for SSB commitment schemes, *oblivious trapdoor generation*, which implies the no-signaling property. This notion is easier to work with in our particular constructions.

Intuitively, this notion captures that there exists a different, *oblivious* key generation algorithm that can generate the commitment key for S and a trapdoor for a subset $S' \subseteq S$ obliviously of $S \setminus S'$ for any subset S' of the larger set S of binding coordinates. More concretely, the oblivious key generation algorithm takes as input a commitment key ck_S binding at S and the description of a subset $S' \subseteq S$ and outputs an *identically distributed key* together with a trapdoor for extracting values in the small set S'. We emphasize that this algorithm does not take as input neither the description of S nor the trapdoor associated with it. Intuitively, the key generation algorithm is oblivious of $S \setminus S'$ (it might even be that $S \setminus S' = \emptyset$) due to the indistinguishability of commitment keys associated with different sets, in this case S and S'.

This property implies no-signaling commitments. Indeed, this follows easily since (1) by the index set hiding property the commitment key itself does not reveal any information about $S \setminus S'$ and (2) we can use the oblivious key generation algorithm to create a trapdoor for extracting the smaller set *without skewing the distribution of the commitment key*. The latter property means essentially that we are given an oracle to extract the smaller set (by computing the trapdoor for an identically distributed key) which is exactly what the no-signaling property captures.

4.1.1.2 Constructing Oblivious SSB Commitments.

We next describe how to construct efficient SSB commitments with oblivious trapdoor generator. A natural way to construct oblivious SSB commitment with locality parameter K is to concatenate K SSB commitments with locality parameter 1. Consider a set $S = \{s_1, \ldots, s_t\}$ for some $t \leq K$. We can construct a commitment key associated with S by computing t commitment keys/trapdoor pairs $(\mathsf{ck}_1, \tau_1), \ldots, (\mathsf{ck}_t, \tau_t)$ for sets $\{s_1\}, \ldots, \{s_t\}$, complement-

ing with $K-t$ keys for $\emptyset$ if necessary. To commit to some $\mathbf{m} \in \mathcal{M}^n$, where $\mathcal{M}$ is the message space of the commitment, one simply computes $\mathsf{c}_1 = \mathsf{Com}_{\mathsf{ck}_1}(\mathbf{m}), \ldots, \mathsf{c}_K = \mathsf{Com}_{\mathsf{ck}_K}(\mathbf{m})$. Extraction of each m_{s_i} is done using c_{s_i} and the trapdoor τ_{s_i}, independently of the others. The oblivious extractor on input the commitment keys for some unknown S and the description of $S' \subseteq S$ just re-samples the commitment keys for S'.[5] Since it doesn't matter if the trapdoors for positions $i \notin S'$ are not known, this trivial extractor can obliviously generate the trapdoor $\{\tau_i : i \in S'\}$.

While this generic construction is enough, we can construct more efficient ones if we consider specific instantiations. More specifically, as we present next, we can have more efficient instantiations (roughly half commitment size compared to the generic one) in the case of commitments derived from the Pedersen commitment scheme.

Efficient SSB Commitments. We next present an oblivious SSB construction based on the Pedersen commitment scheme. This construction was implicit in [GHR15] and later generalized in [FLPS20]. Later we will see that it also satisfies the stronger notion of oblivious trapdoor generation.

Let $\mathbb{G}$ be a group of size p and $\mathbb{F} = \mathbb{Z}_p$. For message space $\mathbb{F}^d$, locality parameter $K \in \mathbb{N}$ and a subset $S \subseteq [d]$ of size $t \leq K$, the commitment key is defined as $\mathbf{G} = (\mathbf{G}_S | \mathbf{G}_{\overline{S}})\mathbf{P}$ where

$$\mathbf{G}_S \leftarrow \mathbb{F}^{(K+1)\times t}, \qquad \mathbf{G}_0 \leftarrow \mathbb{F}^{(K+1)\times(K+1-t)},^{6} \qquad \mathbf{\Gamma} \leftarrow \mathbb{F}^{(K+1-t)\times(d-t)}, \qquad \mathbf{G}_{\overline{S}} = \mathbf{G}_0\mathbf{\Gamma}.$$

Matrix $\mathbf{P} \in \{0,1\}^{d\times d}$ is a permutation matrix associated to S such that $\mathbf{P}\mathbf{e}_{s_i} = \mathbf{e}_i$, for $i \leq t$ and $\mathbf{e}_i$ the i-th vector of the canonical basis. A commitment to $\mathbf{x} \in \mathbb{F}^d$ is computed as

$$[\mathbf{c}] = [\mathbf{G}]\mathbf{x} = [\mathbf{G}_S | \mathbf{G}_{\overline{S}}]\mathbf{P}\mathbf{x} = [\mathbf{G}_S]\mathbf{x}_S + [\mathbf{G}_{\overline{S}}]\mathbf{x}_{\overline{S}}.$$

Note that the columns of $\mathbf{G}_S$ are linearly independent from the columns of $\mathbf{G}_{\overline{S}}$ with overwhelming probability, since $\mathrm{Im}(\mathbf{G}_{\overline{S}}) \subseteq \mathrm{Im}(\mathbf{G}_0)$ and $(\mathbf{G}_S | \mathbf{G}_0)$ is a basis of $\mathbb{F}^{K+1}$ w.o.p. since this corresponds to a uniform matrix of dimensions $K+1 \times K+1$.

This distribution of commitment keys implies that the parts of the input indexed by S go to the space spanned by $\mathbf{G}_S$ of dimension t, while the rest is mapped to the space spanned by $\mathbf{G}_0$ of dimension $K+1-t$. Since $\mathsf{Rank}(\mathbf{G}_S) = t$ with overwhelming probability, all the information of $\mathbf{x}_S \in \mathbb{F}_p^t$ can be retrieved from $\mathbf{c}$. Even more, there exists an efficiently computable trapdoor $\mathbf{T}_S \in \mathbb{F}^{(K+1)\times t}$ such that

$$\mathbf{T}_S^\top \mathbf{G}_S = \mathbf{I}_{t\times t}, \qquad \mathbf{T}_S^\top \mathbf{G}_{\overline{S}} = \mathbf{0}_{t\times(d-t)},$$

[5] Actually, the oblivious key generation needs to know which of the commitments keys $\mathsf{ck}_1, \ldots, \mathsf{ck}_K$ are perfectly binding for $s' \in S'$. Nevertheless, it should be still oblivious of whether the rest of commitment keys are binding or not. See Section 4.3.2 for more details.

[6] It is not always the case that this matrix is uniform. The actual property needed is that this matrix satisfies some hardness assumption. Specifically, the index set hiding property reduces to the $\mathcal{D}$-MDDH assumption where $\mathcal{D}$ is the distributions from which we sample $\mathbf{G}_0$. When working with symmetric groups, we instantiate using the DLin assumption. For the sake of simplicity we consider the uniform case in the technical overview.

and hence

$$\mathbf{T}_S^\top[\mathbf{c}] = \mathbf{T}_S^\top[\mathbf{G}\mathbf{x}] = \mathbf{T}_S^\top[\mathbf{G}_S\mathbf{x}_S + \mathbf{G}_{\overline{S}}\mathbf{x}_{\overline{S}}] = [\mathbf{x}_S].$$

To compute $\mathbf{T}_S$, it is enough to solve the linear system $\mathbf{T}_S^\top(\mathbf{G}_S \mid \mathbf{G}_0) = (\mathbf{I}_S \mid \mathbf{0})$ which admits a solution since $(\mathbf{G}_S \mid \mathbf{G}_0)$ is a basis of $\mathbb{F}^{K+1}$ with overwhelming probability.

Note that this shows also that the commitment is statistically binding in S. The indistinguishability of commitment keys can be shown with a tight reduction to the DDH assumption as in [FLPS20].

Oblivious Trapdoor Generation. One of the main technical contributions of this work is an oblivious trapdoor generator for this commitment scheme, which in turns implies that it is no-signaling. Recall that the property requires that there exists an efficient algorithm, called the oblivious key generation algorithm, that receives as input the description of a set S' of size $t' \leq K$ and a commitment key $[\mathbf{G}]$ sampled for being binding at some unknown $S \supseteq S'$. The algorithm computes a new commitment key $[\mathbf{H}]$ with the following guarantees:

1. it is *statistically close* to $[\mathbf{G}]$ and

2. we also obtain a trapdoor $\mathbf{T}_{S'}$ that allows us to extract local openings for the small set S'.

Since we know that columns in S' are uniformly distributed, we could attempt to sample a uniform matrix $\mathbf{H}_{S'} \leftarrow \mathbb{F}^{(K+1)\times t'}$ and solve the equation $\mathbf{T}_{S'}^\top\mathbf{H}_{S'} = \mathbf{I}_{t'\times t'}$ for some $\mathbf{T}_{S'}$. However, since we don't know the distribution of $[\mathbf{G}_{\overline{S'}}]$, the only hope seems to be to define $[\mathbf{H}_{\overline{S'}}] = [\mathbf{G}_{\overline{S'}}]$ and try to find some $\mathbf{T}_{S'}$ such that $\mathbf{T}_{S'}^\top\mathbf{G}_{\overline{S'}} = \mathbf{0}_{t'\times(d-t')}$. Unfortunately, this amounts to finding elements in the kernel of $[\mathbf{G}_{\overline{S'}}]^\top$ which is in general a computationally hard problem [MRV16].

Instead we make the following observation. Regardless of the distribution of the columns in $S \setminus S'$, the t' lower rows of $\mathbf{G}_{\overline{S}}$ can be always written as a random linear combination of the first $K+1-t'$ rows. That is

$$\mathbf{G}_{\overline{S'}} = \begin{pmatrix} \mathbf{A} \\ \mathbf{R}\mathbf{A} \end{pmatrix}, \text{ where } \mathbf{A} \in \mathbb{F}^{K+1-t'\times d-t'} \text{ and } \mathbf{R} \leftarrow \mathbb{F}^{t'\times K+1-t'}.$$

In this case, if we know the matrix $\mathbf{R}$ in the field, it is possible to compute elements in the kernel of $\mathbf{G}_{\overline{S'}}$ by setting

$$\mathbf{T}_{S'} = \begin{pmatrix} -\mathbf{R}^\top\mathbf{C} \\ \mathbf{C} \end{pmatrix}, \text{ for any } \mathbf{C} \in \mathbb{F}^{t'\times t'}.$$

If additionally, we choose some $\mathbf{C}$ that satisfies $\mathbf{T}_{S'}^\top\mathbf{H}_{S'} = \mathbf{I}_{t'\times t'}$ we have computed a trapdoor for S'. This yields a way to compute the rest of the columns: discard the lower t' rows of $\mathbf{G}_{\overline{S}}$, sample a uniform matrix $\mathbf{R}$ as above and complete the last rows with the elements $\mathbf{R}[\mathbf{A}]$. Then, using $\mathbf{R}$, $\mathbf{H}_{S'}$ (which are known in the field) find some $\mathbf{C}$ that satisfies the linear equations and use it to define the trapdoor $\mathbf{T}'_S$.

Let's see in more detail why the previous observation holds. Consider the matrix $\mathbf{G}_0 \in \mathbb{F}^{(K+1)\times(K+1-t)}$ and note that the upper part $\overline{\mathbf{G}}_0$ is a uniformly distributed matrix with more rows than columns; hence $\mathbf{R}\overline{\mathbf{G}}_0$, for $\mathbf{R} \leftarrow \mathbb{F}^{t'\times(K+1-t')}$, is uniformly distributed. This is also valid for all non-binding coordinates since $\mathbf{G}_{\overline{S}} = \mathbf{G}_0\mathbf{\Gamma}$ and then the lower rows follow distribution $\mathbf{R}\overline{\mathbf{G}}_{\overline{S}}$. Next, consider the columns corresponding to the (unknown) binding coordinates $S \setminus S'$. The same argument holds: for some uniform $\mathbf{R}'\overline{\mathbf{G}}_{S\setminus S'}$ is uniform when $\mathbf{R}' \leftarrow \mathbb{F}^{t'\times(K+1-t')}$. It remains to show that using the same randomness for both column sets, i.e. setting $\mathbf{R} = \mathbf{R}'$, does not alter the distribution of the commitment key. Indeed, with overwhelming probability, the columns of $\overline{\mathbf{G}}_0 \in \mathbb{F}^{(K+1-t')\times(K+1-t)}$ and of $\overline{\mathbf{G}}_{S\setminus S'} \in \mathbb{F}^{(K+1-t')\times(t-t')}$ form a basis of $\mathbb{F}^{K+1-t'}$, which means that the matrix $\mathbf{R}^\top$ can be decomposed into two independent components: a random element in $\mathrm{Im}(\overline{\mathbf{G}}^{\perp}_{S\setminus S'})$ and another in $\mathrm{Im}(\overline{\mathbf{G}}^{\perp}_0)$. This shows that

$$\mathbf{R}\overline{\mathbf{G}}_0 = \mathbf{R}_2(\mathbf{G}^{\perp}_{S\setminus S'})^\top\overline{\mathbf{G}}_0, \qquad \mathbf{R}\overline{\mathbf{G}}_{S\setminus S'} = \mathbf{R}_1(\mathbf{G}^{\perp}_0)^\top\overline{\mathbf{G}}_{S\setminus S'}$$

are independent and therefore $\begin{pmatrix} \overline{\mathbf{G}}_{S\setminus S'} & \overline{\mathbf{G}}_0\mathbf{\Gamma} \\ \mathbf{R}\overline{\mathbf{G}}_{S\setminus S'} & \mathbf{R}\overline{\mathbf{G}}_0\mathbf{\Gamma} \end{pmatrix}$ is correctly distributed.

4.1.2 Pairing-based Quasi-Arguments

Paneth and Rothblum [PR17] and then Kalai et al. [KPY19] used a weakened version of an argument of knowledge called quasi-argument, as an intermediate step for obtaining a delegation scheme. Quasi arguments are defined for languages that can be expressed as a set of *local constraints*. Roughly speaking, this means that a witness $\mathbf{w}$ for membership of a statement x in a language can be decomposed in parts, namely $\mathbf{w} = (\mathsf{w}_1, \ldots, \mathsf{w}_n)$, and for each subset $S \subseteq [n]$, the partial witness $\mathbf{w}_S$ satisfies some local relations, that is, a predicate $\mathcal{R}(\mathsf{x}, \mathbf{w}_S)$ holds. For example, in the case of a CNF formula of n variables, the witness is an accepting assignment of the formula and a local constraint with respect to some set S captures that every clause that only has variables $\mathsf{w}_i, \mathsf{w}_j, \mathsf{w}_k$ for $i, j, k \in S$ is satisfied. Note that it can be the case that even unsatisfiable formulas can satisfy all local constraints for families of sets of small size (yet, no global satisfying assignment exists).

Unlike an argument of knowledge, a quasi-argument has only local extraction, meaning that only a small part of the witness of size at most K, the locality parameter, is extracted. This is formalized by means of an extractor which on input a set $S \subseteq [n]$ of size at most K, where n is the size of the witness, programs an srs so that it can later extract positions of the witness defined by S. Central to quasi-arguments is the notion of no-signaling local extraction which is aimed to capture a strong *local soundness* guarantee.

Local soundness requires that the extracted local witness is consistent with the relation and doesn't lead to a local contradiction, that is, it satisfies the local constraints associated to some set S. The *no-signaling* requirement is defined for any two sets S, S' where $S' \subseteq S$ and of size at most K. It states that the result of programming extraction for S and then output only the extracted value for S', should be indistinguishable from the result of programming

extraction for S' and output the extracted value for S'. Intuitively, this strengthens locality by requiring that the small parts of the local witness are extracted independently.

We next outline the construction of pairing-based quasi-arguments for two specific languages of interest, satisfiability of linear and quadratic relations on committed values. For ease of presentation we do so for symmetric bilinear groups but we stress out that we also translate these to the more efficient setting of asymmetric bilinear groups. We will later rely on these quasi arguments to construct a delegation scheme for polynomial sized arithmetic circuits but we emphasize that these constructions are of independent interest; they capture a form of "succinct" aggregation of relations and -importantly- they do so under standard falsifiable assumptions. While full knowledge soundness is not achieved, the weakened notion of no-signaling extraction might be enough for some applications. Thus, we choose to present them in full generality.

Before presenting the quasi arguments, we briefly recall a few constructions on which we build on: the QA-NIZK construction for membership in linear spaces of [KW15] and the knowledge transfer arguments introduced in [GR19] which allow to construct QA-NIZK under falsifiable assumptions in some more restricted setting.

Quasi-Adaptive NIZK for membership in linear spaces. Quasi-Adaptive NIZK (QA-NIZK)[7] arguments are NIZK arguments where the srs is allowed to depend on the specific language for which proofs have to be generated [JR13]. We are interested in the specific language of membership in linear spaces. Specifically, given a matrix $\mathbf{M}$ and a description of a group gk, we consider the language of vectors of group elements that lie in the image of $\mathbf{M}$, that is,

$$\mathcal{L}_{\mathrm{gk},\mathbf{M}} = \{[\mathbf{x}] \mid \exists \mathbf{w} \text{ s.t. } \mathbf{x} = \mathbf{M}\mathbf{w}\}$$

In the quasi-adaptive model, we allow the reference string to depend on gk and $\mathbf{M}$, but an adversary can choose the statement $[\mathbf{x}]$ adaptively.

There are very efficient constructions in this setting. We briefly describe the construction of Kiltz and Wee [KW15]. First we consider the designated verifier case. Let $\mathbf{M}$ be an $\ell \times n$ matrix. The construction is essentially a hash proof system [CS02]. The srs contains the projection $[\mathbf{B}] = [\mathbf{M}^\top \mathbf{K}]$ for a random secret key $\mathbf{K} \in \mathbb{F}^{\ell \times k}$. To prove a statement $[\mathbf{x}] = [\mathbf{M}]\mathbf{w}$, the prover sends $[\boldsymbol{\pi}] = \mathbf{w}^\top[\mathbf{B}]$ and the verifier asserts that $[\boldsymbol{\pi}] = [\mathbf{x}]^\top \mathbf{K}$. Now it is easy to see that this simple protocol is complete. Indeed

$$\boldsymbol{\pi} = \mathbf{w}^\top[\mathbf{B}] = \mathbf{w}^\top \mathbf{M}^\top \mathbf{K} = \mathbf{x}^\top \mathbf{K}$$

For soundness, roughly speaking, the value $\mathbf{x}^\top \mathbf{K}$ is random for $\mathbf{x}$ that does not belong to the image of $\mathbf{M}$ conditioned on $\mathbf{B}$. Thus, a cheating (even unbounded) prover has only negligible probability of producing a verifying proof for elements not in the image of $\mathbf{M}$.

[7] In this work we do not need the zero knowledge property so we omit it from the discussion.

To make the scheme publicly verifiable, groups equipped with a bilinear map are employed. To enable the verifier to perform the verification test without knowing the secret $\mathbf{K}$, we also add to the srs the value $[\mathbf{C}] = [\mathbf{KA}]$, where $\mathbf{A}$ is a matrix that satisfies some hardness condition. Now, the verifier can test

$$e([\boldsymbol{\pi}], [\mathbf{A}]) = e\left([\mathbf{x}^\top], [\mathbf{C}]\right).$$

Note that this corresponds to multiplying the verification equation of the designated verifier case from the right with $\mathbf{A}$. Now, if

(1) the designated verifier relation does not hold, namely, $\boldsymbol{\pi} \neq \mathbf{x}^\top \mathbf{K}$ and

(2) the proof verifies, namely $\boldsymbol{\pi}\mathbf{A} = \mathbf{x}^\top \mathbf{KA}$,

then $[\boldsymbol{\pi}] - [\mathbf{x}^\top]\mathbf{K}$ is a non-trivial element in the co-kernel of $[\mathbf{A}]$. Thus, the publicly verifiable scheme is sound if we additionally assume that $\mathbf{A}$ is sampled by a distribution $\mathcal{D}$ such that the $\mathcal{D}$-Kernel Diffie-Hellman assumption holds.

Note that if $\mathbf{M}$ spans the entire linear space, then the language is trivial. In this case, only knowledge soundness is a meaningful property. However, we do not know whether knowledge soundness of this construction can be proven under falsifiable assumptions or not.

Knowledge Transfer Arguments. To achieve succinct arguments, in principle, one needs to use shrinking commitments. When trying to use such commitments with QA-NIZK such as [KW15], the aforementioned "triviality" problem arises and it seems like one has to resort to non-falsifiable assumptions or the generic group model. Motivated by the problem of constructing delegation schemes under falsifiable assumptions and in order to overcome the above issue, [GR19] relax the knowledge soundness property.

When considering delegation using the natural approach of (deterministically) committing to the wires of the circuit, one can observe that full knowledge soundness seems to be an unnecessarily strong requirement. Indeed, given the input $\mathbf{x}$ of the circuit, one can compute (or verify) these commitments efficiently by evaluating the circuit. This means intuitively, that we already know how a "correct" opening of the commitments looks like in the soundness security reduction. [GR19] exploits this fact and manages to relax the knowledge soundness requirement by considering statements of the form "if commitment $[\mathbf{c}]$ opens to $\mathbf{w}$, then commitment $[\mathbf{d}]$ opens to $f(\mathbf{w})$" for a publicly known function f. As we shall see later, they show that this notion of soundness is enough to construct delegation for low-depth circuits. They also construct two knowledge transfer arguments for linear and quadratic relations under falsifiable assumptions. More concretely, they consider statements of the form

- "if $[\mathbf{c}]$ opens to $\mathbf{Mw}$, then $[\mathbf{d}]$ opens to $\mathbf{Nw}$ for some publicly known $\mathbf{M}, \mathbf{N}$, and
- "if $[\mathbf{c}_1]$ opens to $\mathbf{w}_1$ and $[\mathbf{c}_2]$ opens to $\mathbf{w}_2$, then $[\mathbf{d}]$ opens to $\mathbf{w}_1 \circ \mathbf{w}_2$.

In the soundness definition, the adversary is required to output the valid opening along with the statement/proof pair. We emphasize that this is only part of the soundness definition

and in the protocol execution the prover does not have to output the valid opening. Consider for example the first case for linear relations. An adversary wins if it manages to output a statement $[\mathbf{c}], [\mathbf{d}]$ with an accepting proof *and* $\mathbf{w}$ such that $[\mathbf{c}] = [\mathbf{M}]\mathbf{w}$ *but* $[\mathbf{d}] \neq [\mathbf{N}]\mathbf{w}$. Such statements essentially give the guarantee that some a priori knowledge about a commitment is "correctly" transferred to another commitment.

For the former construction, namely linear relations, they use the [KW15] construction where they define $\mathbf{M}$ as a two block matrix where the upper part corresponds to $[\mathbf{c}]$ and the lower to $[\mathbf{d}]$. Now, using [KW15], the prover simply needs to convince the verifier that

$$\begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix} = \begin{bmatrix} \mathbf{M} \\ \mathbf{N} \end{bmatrix} \mathbf{w}.$$

They show that this construction is knowledge transfer sound if the upper matrix $\mathbf{M}$ is sampled from a distribution $\mathcal{D}$ for which the $\mathcal{D}$-MDDH assumption holds.

For proving the quadratic relations, they do a different analysis of standard techniques used for the construction of pairing-based succinct arguments that exploit the properties of the Lagrange basis.

They also modify these constructions to be compatible with the more efficient setting of asymmetric bilinear groups, under the natural modifications of the required assumption for asymmetric setting.

4.1.2.1 Oblivious Trapdoor Generation for Quasi-Arguments

Similar to the case of no-signaling SSB commitments we define a stronger and easier to work with (in our context) notion that implies the no-signaling property of quasi arguments, *oblivious trapdoor generation*.

We require that there exists an *oblivious* key generation algorithm that takes as input (1) an srs_S that allows extraction for a set S, and (2) the description of a subset $S' \subseteq S$, and generates a $\mathsf{srs}_{S'}$ for some set S' *and* a trapdoor[8] for extracting local witnesses associated to the set S' *obliviously* of $S \setminus S'$. We emphasize that the oblivious trapdoor generation algorithm knows neither the description of S nor any information about the trapdoor associated with it. We require that the new srs is *statistically close* to the srs_S given as input. The fact that this property implies no-signaling commitments is identical to the case of SSB commitments.

4.1.2.2 Quasi-Arguments of Membership in a Linear Space

We define a quasi-argument of knowledge of some vector $[\mathbf{x}] \in \mathbb{G}^\ell$ belonging to the image of a matrix $[\mathbf{U}] \in \mathbb{G}^{\ell \times n}$, where $\mathbf{x}$ is committed using an SSB commitment. Consider a commitment

[8] We modify the quasi-argument definition of [KPY19] to admit a fixed extractor algorithm that takes as input the statement-proof pair of the adversary, and additionally some secret state produced during the srs generation, -the trapdoor- and extracts the local witness.

$[\mathbf{c}]$ that is statistically binding on the set S. We show that there exists a local and no-signaling extractor which, given some $S \subseteq [n]$ of size $t \leq K$, extracts $[\mathbf{x}_S] \in \mathrm{Im}([\mathbf{U}_S])$, where $\mathbf{x}_S \in \mathbb{F}^t$ is the vector whose entries are x_i and $\mathbf{U}_S \in \mathbb{F}^{t \times n}$ is the matrix whose rows are the rows of $\mathbf{U}$ indexed by i, where i ranges over S in some fixed order. A local constraint $[\mathbf{x}_S]$ associated with the set S can be interpreted as satisfying two properties:

1. $[\mathbf{x}_S]$ is consistent with the commitment $[\mathbf{c}]$, namely the (unique) S-opening of $[\mathbf{c}]$ is $\mathbf{x}_S$, and
2. $[\mathbf{x}_S]$ is in the image of $[\mathbf{U}_S]$.

We use the Kiltz and Wee argument of membership in linear spaces [KW15] to construct a quasi argument for linear relations. Details follow.

The argument. Our construction is Kiltz and Wee linear membership argument [KW15] for the matrix $[\mathbf{GU}]$, where $\mathbf{G}$ is an SSB commitment key with locality parameter K. For completeness, we describe the protocol for this specific matrix. We note that we present the scheme with proof size $k+1$ of [KW15], where k is a parameter of the scheme defined by the underlying assumption, but our construction is also sound for the more efficient instantiation of size k. In any case, we emphasize that the parameter is a small constant ($k = 2$).

Let's recall the construction for the matrix $\mathbf{M} = \mathbf{GU}$. The srs contains $[\mathbf{B}] = [\mathbf{U}^\top \mathbf{G}^\top \mathbf{K}]$ and $[\mathbf{C}] = [\mathbf{KA}]$ for some random hash key $\mathbf{K}$ and $\mathbf{A}$ drawn from some distribution satisfying a kernel assumption. A proof is computed as $[\boldsymbol{\pi}] = \mathbf{w}^\top [\mathbf{B}]$, and verification is done by checking if $e([\pi], [\mathbf{A}]) = e([\mathbf{c}^\top], [\mathbf{C}])$.

Local and No-Signaling extraction. Our strategy to prove local soundness is to show that, apart from extracting $[\mathbf{x}_S]$ from $[\mathbf{c}]$, we are also able to produce a verifying proof $[\boldsymbol{\pi}^\dagger]$ that $[\mathbf{x}_S] \in \mathrm{Im}(\mathbf{U}_S)$. More concretely, on input $\mathsf{srs}_S = ([\mathbf{A}^\dagger], [\mathbf{B}^\dagger], [\mathbf{C}^\dagger])$ for membership in the linear space of $\mathbf{U}_S$, we can construct another srs that is statistically close to the quasi argument srs for $\mathbf{U}$ and, more importantly, we can extract a local opening $[\mathbf{x}_S]$ *and* a proof $[\boldsymbol{\pi}]^\dagger$ satisfying the verification equation for srs_S.

We embed the public parameters $[\mathbf{A}^\dagger], [\mathbf{B}^\dagger], [\mathbf{C}^\dagger]$ of the local linear space argument for $\mathbf{U}_S$ in the quasi argument parameters. Although the secret hash key $\mathbf{K}^\dagger$ of the local linear argument is statistically hidden, we can still pick a random hash key for all the coordinates by picking another secret key and implicitly define the full secret key as some composition of the two keys. Concretely, given the trapdoor $\mathbf{T}_S$ for locally opening SSB commitments we implicitly define $\mathbf{K} = \mathbf{T}_S \mathbf{K}^\dagger + \mathbf{R}$, where $\mathbf{R}$ is the additional key, so that the proofs for

$$\mathbf{c} = \mathbf{GP} \begin{pmatrix} \mathbf{x}_S \\ \mathbf{x}_{\overline{S}} \end{pmatrix} = \mathbf{G}_S \mathbf{x}_S + \mathbf{G}_{\overline{S}} \mathbf{x}_{\overline{S}}$$

are of the form

$$\boldsymbol{\pi} = \mathbf{c}^\top \mathbf{K} = (\mathbf{G}_S \mathbf{x}_S + \mathbf{G}_{\overline{S}} \mathbf{x}_{\overline{S}})^\top (\mathbf{T}_S \mathbf{K}^\dagger + \mathbf{R}) = \mathbf{x}_S^\top \mathbf{K}^\dagger + \mathbf{c}^\top \mathbf{R}.$$

In this way, a proof for the local argument can be retrieved as $[\boldsymbol{\pi}]^\dagger = [\boldsymbol{\pi}] - [\mathbf{c}^\top]\mathbf{R}$. This equivalent way of sampling $\mathbf{K}$ allows to compute the srs of the larger linear argument using only $[\mathbf{A}^\dagger], [\mathbf{B}^\dagger], [\mathbf{C}^\dagger]$ and $\mathbf{T}_S, \mathbf{R}$. Indeed, we can define $[\mathbf{A}] = [\mathbf{A}^\dagger]$, $[\mathbf{B}] = [\mathbf{B}^\dagger] + [\mathbf{U}^\top\mathbf{G}^\top]\mathbf{R}$ and $[\mathbf{C}] = \mathbf{T}_S[\mathbf{C}^\dagger] + \mathbf{R}[\mathbf{A}^\dagger]$.

We also show that the srs is indistinguishable for different sets and that there is an oblivious trapdoor generation strategy, and hence we also have a no-signaling extraction strategy. The indistinguishability of the srs follows directly from the indistinguishability of SSB commitment keys; it is enough to note that only the commitment key depends on S and all other values can be efficiently computed given only the commitment key[9]. For oblivious trapdoor generation, we use the fact that we can sample an identically distributed commitment key along with a trapdoor -this follows by the oblivious key generation of the commitment scheme- and then we argue in the same way as before: given the commitment key we can sample the rest of srs honestly.

Extension to Knowledge Transfer, Bilateral Spaces and Sum Arguments. We also construct variations of the above protocol, specifically a knowledge transfer version based on [GR19] and two construction suitable for asymmetric bilinear groups.

First we consider the knowledge transfer construction. We first describe the local constraints. Consider two matrices $[\mathbf{M}], [\mathbf{N}]$, and two commitment keys $[\mathbf{G}], [\mathbf{H}]$ statistically binding at S. The statement consists of two commitments $[\mathbf{c}], [\mathbf{d}]$. For the local extraction guarantee w.r.t. set S we require that, given an accepting proof π *and* an opening $\mathbf{w}$, we can extract values $[\mathbf{x}_S]$, $[\mathbf{y}_S]$ such that

- $[\mathbf{x}_S]$, $[\mathbf{y}_S]$ are the unique S-openings of $[\mathbf{c}], [\mathbf{d}]$ w.r.t. commitment keys $\mathbf{G}, \mathbf{H}$ respectively, and
- if $[\mathbf{x}_S] = [\mathbf{M}_S]\mathbf{w}$, then $[\mathbf{y}_S] = [\mathbf{N}_S]\mathbf{w}$.

The construction and the analysis are identical to the previous case. We use the [KW15] construction for the matrix with upper part $\mathbf{GM}$ and lower part $\mathbf{HN}$. The only difference in the analysis is on the local extraction case. We argue that we can extract an accepting proof for an srs for the language of linear knowledge transfer for the matrices $\mathbf{M}_S, \mathbf{N}_S$ and, thus, we also require that the $\mathcal{M}_S^\top$-MDDH assumption holds for every S, where $\mathcal{M}_S$ is the distribution from which we sample $\mathbf{M}_S$.

Finally, we also consider constructions in asymmetric bilinear groups. A variant of the linear subspace QA-NIZK argument given in [GHR15], and extended to knowledge transfer arguments in [GR19], considers the statement as well as the matrix split between the two groups. We call this argument a linear argument for bilateral spaces. We also consider a particular type of argument for bilateral linear spaces defined in [GHR15] and called "sum in subspace argument".

[9]Here, we assume the distribution $\mathcal{U}$ that outputs the matrix $[\mathbf{U}]$ is witness samplable, meaning that during sampling, we can also sample the discrete logarithms of $[\mathbf{U}]$ which is usually the case. In this work, we only consider such distributions.

In this case, the statement is $[\mathbf{x}]_1, [\mathbf{y}]_2$ and soundness captures that $\mathbf{x}+\mathbf{y} \in \mathrm{Im}(\mathbf{M}+\mathbf{N})$ given $[\mathbf{M}]_1, [\mathbf{N}]_2$ in the two different source groups. We construct quasi arguments for all these variants with knowledge transfer soundness. Luckily, the constructions as well as the security proofs are minor modifications of the original argument.

4.1.2.3 Quasi-Argument of Hadamard Products

The next quasi argument construction shows that some vector $\mathbf{c}$ is the Hadamard product of two vectors $\mathbf{a}, \mathbf{b}$, namely $\mathbf{c} = \mathbf{a} \circ \mathbf{b}$. We can naturally define the local constraints here as $\mathbf{c}_S = \mathbf{a}_S \circ \mathbf{b}_S$ for every set $S \subseteq [n]$, where n is the dimension of the vectors. As in the linear case, we care about committed values, that is, the vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are committed and we claim that the openings satisfy the claimed relation.

Our starting point is the "bit-string" argument of [GHR15]. We observe that it is implicitly a quasi-argument with locality parameter $K = 1$ for the set of equations $b_i(b_i - 1) = 0$ for all $i \in [n]$. Next we describe this construction and then we show it indeed satisfies the no-signaling local soundness property. It will be convenient to directly work with equations of the form $x_i y_i = z_i$ instead of the bit-string argument equations.

The common reference string in [GHR15] contains what we interpret as three SSB commitment keys $[\mathbf{G}], [\mathbf{H}], [\mathbf{F}]$ with locality parameter $K = 1$. It additionally includes the product $[\mathbf{G} \otimes \mathbf{H}]$. The prover gives three commitments $[\mathbf{a}], [\mathbf{b}], [\mathbf{c}]$ w.r.t. $\mathbf{G}, \mathbf{H}, \mathbf{F}$ and claims that the openings satisfy the Hadamard relation. We first note that it is easy to construct an argument for a related language. Consider the elements $\mathbf{G} \otimes \mathbf{H}$ as a commitment key. The prover can give a commitment to the Kronecker product $\mathbf{z} = \mathbf{a} \otimes \mathbf{b}$ by computing $[\mathbf{t}] = [\mathbf{G} \otimes \mathbf{H}]\mathbf{z}$. The verifier can then use the pairing to verify the Kronecker product relation, namely it tests that $e([\mathbf{c}], [\mathbf{d}]) = e([\mathbf{t}], [1])$ where $[\mathbf{c}] = [\mathbf{G}]\mathbf{a}, [\mathbf{d}] = [\mathbf{H}]\mathbf{b}$ are commitment to some vectors and are part of the statement. Some simple calculations show that

$$\mathbf{cd} = \mathbf{c} \otimes \mathbf{d} = \mathbf{Ga} \otimes \mathbf{Hb} = (\mathbf{G} \otimes \mathbf{H})(\mathbf{a} \otimes \mathbf{b}) = \mathbf{t}$$

The Kronecker product commitment $\mathbf{t}$ is included as part of the proof. Now, from this simple Kronecker product argument, it is easy to prove the Hadamard product. It is enough to note that the Hadamard product is a linear function of the Kronecker product, thus, the prover and verifier can use the protocol for linear relations of the previous section.

Local and No-Signaling Extraction. The crucial observation to prove local extraction is that if $\mathbf{G}, \mathbf{H}$ are extractable in one position, say i, j respectively, then $\mathbf{G} \otimes \mathbf{H}$ is extractable at position $n(i-1)+j$. More concretely, letting $\mathbf{T_G}$, $\mathbf{T_H}$ be the trapdoors for $\mathbf{G}, \mathbf{H}$ respectively, the trapdoor for the commitment key $\mathbf{G} \otimes \mathbf{H}$ is simply $\mathbf{T_G} \otimes \mathbf{T_H}$. Some straightforward calculations reveal that applying this trapdoor to a commitment with the key $\mathbf{G} \otimes \mathbf{H}$ indeed yields the $n(i-1)+j$-th coordinate of the committed value, which is uniquely defined. In fact, we generalize this for larger locality parameters and we also show that, for some distributions of commitment keys, the no-signaling/oblivious trapdoor generation properties hold if they hold for $\mathbf{G}$, $\mathbf{H}$.

Consider the simple case of $K = 1$ and let all three commitments $\mathbf{G}, \mathbf{H}, \mathbf{F}$ be extractable at the same position i. We show that we can extract local openings $[x_i] = \mathbf{T_G}[\mathbf{a}], [y_i] = \mathbf{T_H}[\mathbf{b}], [z_i] = \mathbf{T_F}[\mathbf{c}]$ as well as $[w_i] = \mathbf{T_{G\otimes H}}[\mathbf{t}]$ such that $z_i = x_i y_i$. Assume for the sake of a contradiction that $z_i \neq z'_i = a_i b_i$. Since the columns $\mathbf{g}_i, \mathbf{h}_i, \mathbf{f}_i$ are linearly independent from the other columns in $\mathbf{G}, \mathbf{H}, \mathbf{F}$, respectively, if the commitments $[\mathbf{c}], [\mathbf{d}], [\mathbf{t}]$ satisfy $[\mathbf{c}] \otimes [\mathbf{d}] = e([\mathbf{t}], [1])$, then the unique openings at coordinate i satisfy $z_i = x_i y_i$. Now, if $z_i \neq z'_i$, the linear relation does not hold and we can break the underlying QA-NIZK for membership in linear spaces.

For oblivious trapdoor generation, it is enough to note that if the commitment key satisfies this property, so does the above constructions. Indeed, note that using the commitment key, it is enough to produce an srs for membership in subspace language to create the full srs of the protocol.

Extension to Knowledge Transfer Arguments. We extend the quasi-argument local soundness to offer a "knowledge transfer" guarantee. In this case, we essentially commit to commitments. That is, we use an SSB commitment key to commit to multiple commitments and the local openings are commitments themselves. Namely we extract values $[x_i], [y_i], [z_i]$ which are interpreted as commitments w.r.t. some (not necessarily SSB) commitments keys $\mathbf{U}, \mathbf{V}, \mathbf{W}$. We require that no PPT adversary can produce openings $\mathbf{a}, \mathbf{b}$ such that $x_i = \mathbf{U}_i\mathbf{a}, y_i = \mathbf{V}_i\mathbf{b}$ but $z_i \neq \mathbf{W}_i\mathbf{a} \circ \mathbf{b}$. The constraint language for a set S is parameterized by SSB commitments $\mathbf{G}, \mathbf{H}, \mathbf{F}$ binding at S as well as some matrices $\mathbf{U}, \mathbf{V}, \mathbf{W}$. We require that given an accepting proof π for a statement $[\mathbf{c}], [\mathbf{d}], [\mathbf{f}]$ *and* openings $\mathbf{a}, \mathbf{b}$, we can extract values $[\mathbf{x}_S]$, $[\mathbf{y}_S]$, $[\mathbf{z}_S]$ such that

1. $[\mathbf{x}_S]$, $[\mathbf{y}_S]$, $[\mathbf{z}_S]$ are the unique S-openings of $[\mathbf{c}]$, $[\mathbf{d}]$, $[\mathbf{f}]$ w.r.t. commitment keys $\mathbf{G}, \mathbf{H}, \mathbf{F}$ respectively, and

2. if $[\mathbf{x}_S] = [\mathbf{U}_S]\mathbf{a}$ and $[\mathbf{y}_S] = [\mathbf{V}_S]\mathbf{b}$, then $[\mathbf{z}_S] = [\mathbf{W}_S]\mathbf{a} \circ \mathbf{b}$.

One might wonder at this point how we commit to commitments which naturally requires multiplication of group elements which is assumed computationally hard. To achieve that, we simply include in the srs the products $[\mathbf{GU}], [\mathbf{HV}], [\mathbf{FW}]$. Now, we can commit to the n commitments $\mathbf{U}_i\mathbf{a}$ as $[\mathbf{GU}]\mathbf{a}$ and similarly for the other keys.

The knowledge transfer version is essentially the same as in the previous case. The only difference is that we also need to include some additional elements in the srs to allow to the prover to compute the Kronecker product, namely the values $[\mathbf{Q}] = [(\mathbf{G}\otimes\mathbf{H})(\mathbf{U}\otimes\mathbf{V})]$. As in the previous case, we can then exploit the linear relation between the Hadamard product and the Kronecker product. From a correct commitment $[\mathbf{Q}](\mathbf{a} \otimes \mathbf{b})$, we can use the linear knowledge transfer to get a commitment to the Hadamard products w.r.t. the third commitment key, namely $[\mathbf{FW}](\mathbf{a} \circ \mathbf{b})$. To show this, we first show that the $\mathcal{G} \otimes \mathcal{H}$-MDDH assumption holds if $\mathcal{G}$-MDDH and $\mathcal{H}$-MDDH hold, where $\mathcal{G}, \mathcal{H}$ are the distributions of $\mathbf{G}, \mathbf{H}$ respectively.

We are also able to extend these techniques to work in asymmetric bilinear groups as well. The construction is somewhat technical, but the core idea is to construct SSB commitments

suitable for asymmetric groups, where we "split" the commitments between the two groups, and use the bilateral variants of the linear quasi-arguments discussed in the previous sections.

4.1.3 From our Quasi-Arguments to Delegation.

Using the ideas of [PR17; KPY19], we can derive delegation of computation from quasi arguments for languages encoding the computation. The local constraints capture that each step of the computation was done correctly. First, we present the high level idea for the delegation construction from quasi-arguments. We first show how to delegate low-space TMs/low-width circuits and then we show how to overcome the dependence on space/width.

4.1.3.1 Delegating bounded space TM/bounded width circuits

We first recall the high-level ideas to construct a delegation scheme from quasi arguments of [PR17; KPY19] in the simpler case of bounded space computation. Consider some polynomial time sequential computation which on input x outputs y, for example a Turing Machine or an arithmetic circuit. The computation goes through a sequence of states $\mathtt{st}_0, \mathtt{st}_1, \ldots, \mathtt{st}_d$ such that $\mathtt{st}_0$ is consistent with the input, state $\mathtt{st}_d$ contains the output y, and there's a functional relation between states $\mathtt{st}_i, \mathtt{st}_{i+1}$ where $\mathtt{st}_{i+1} = f(\mathtt{st}_i)$ and f is determined by the description of the computation. We first consider the case of bound space computation and discuss later how to remove this constraint. Consider a quasi argument of locality $K = 2|\mathtt{st}|$ where local constraints require that $\mathtt{st}_i, \mathtt{st}_{i+1}$ are consistent w.r.t. f. The goal is to show that an adversary that makes the quasi-argument verifier accept must (w.o.p) sample x, y such that y is the result of the computation on input x.

We can first "program" the local extractor extractor to extract $\mathtt{st}_0, \mathtt{st}_1$, i.e. use locality parameter $K = 2|\mathtt{st}|$, where $|\mathtt{st}|$ is a bound on the size of the states (i.e. space of the TM or width of the circuit). Local soundness asserts that state $\mathtt{st}_0$ is consistent with x. Local soundness also implies that $\mathtt{st}_1$ is consistent with $\mathtt{st}_0$ and hence with x (note that the statement $\mathtt{st}_1 = f(\mathtt{st}_0)$ depends only on local variables). Now, to show that $\mathtt{st}_2$ is also consistent, we jump to another game where first the extractor computes only $\mathtt{st}_1$, and in the next game the extractor computes $\mathtt{st}_1, \mathtt{st}_2$. The crucial observation is that $\mathtt{st}_1$ should be still consistent with x in both games. Otherwise, we can distinguish between the common output of extractors for $\mathtt{st}_0, \mathtt{st}_1$ and $\mathtt{st}_1$ or between $\mathtt{st}_1$ and $\mathtt{st}_1, \mathtt{st}_2$, which contradicts the no-signaling property. Importantly, we can efficiently compute the "correct" state $\mathtt{st}_1$ since the computation is deterministic, and thus the no-signaling distinguisher described is indeed efficient. Similarly, consistency of $\mathtt{st}_1$ and local soundness imply that $\mathtt{st}_2$ is also consistent. Now, we can inductively continue until we reach the last state, $\mathtt{st}_d$, which corresponds to the output of the computation.

Small width circuit delegation from DLIN. Let C be an arithmetic circuit with width w and depth d. We consider the input to correspond to level 0. Without loss of generality, assume that the circuit has w input and w output wires. In this section we consider the width

w to be small, or alternatively, efficiency will depend on w.

We follow the circuit arithmetization of [GR19]. The multiplication gates are partitioned in d levels. Each level groups the gates at the same distance from the inputs, without counting linear gates. In this way, the inputs of level $i+1$ are linear combinations of outputs of the i previous levels. We can then express this as constraints describing the computation as

$$\mathbf{a}_i \circ \mathbf{b}_i = \mathbf{c}_i \qquad \text{for } i = 1 \text{ to } d, \tag{4.1}$$

$$\begin{pmatrix}\mathbf{a}_{i+1}\\ \mathbf{b}_{i+1}\end{pmatrix} = \sum_{0\le j\le i}\begin{pmatrix}\mathbf{D}_{i,j}\\ \mathbf{E}_{i,j}\end{pmatrix}\mathbf{c}_j = \begin{pmatrix}\mathbf{D}_i & \mathbf{0}\\ \mathbf{E}_i & \mathbf{0}\end{pmatrix}\mathbf{c} \qquad \text{for } i = 0 \text{ to } d-1, \tag{4.2}$$

$$\mathbf{c}_0 = \mathbf{x} \in \mathbb{F}^w \text{ and } \mathbf{c}_d = \mathbf{y} \in \mathbb{F}^w. \tag{4.3}$$

Vectors $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i$ denote respectively the left, right and output wires of multiplication gates in level i. Matrices $\mathbf{D}_{i,j}, \mathbf{E}_{i,j}$ can be naturally derived from the circuit's linear gates. Equation 4.1 states the relation between output wires and the input wires of a level of multiplication gates.

Now consider a symmetric bilinear group described by gk and consider three SSB commitments $\mathbf{G}, \mathbf{H}, \mathbf{F}$ with locality $K = |w|$ for committing to wd-dimensional vectors. We publish in the srs the commitment keys and we we also compute two quasi argument srs:

1. for membership in linear space for the matrix $[\mathbf{M}_1] = \begin{bmatrix}\mathbf{F}\\ \mathbf{GD}\\ \mathbf{HE}\end{bmatrix}$. Here, $\mathbf{D}$, $\mathbf{E}$ are the matrices for the linear relations as a whole (note per level). That is, for left and output wires it should hold $\mathbf{a} = \mathbf{Dc}$, and similarly for right wires.
2. for Hadamard relation for $\mathbf{G}, \mathbf{H}, \mathbf{F}$. Note that, essentially, this corresponds to yet another quasi argument for membership in linear spaces for $[\mathbf{M}_2] = \begin{bmatrix}(\mathbf{G}\otimes\mathbf{H})\\ \mathbf{F\Delta}\end{bmatrix}$ where $\mathbf{\Delta}$ captures the linear relation between the Kronecker and Hadamard product, that is $(\mathbf{a}\circ\mathbf{b}) = \mathbf{\Delta}(\mathbf{a}\otimes\mathbf{b})$.

The prover gives the commitments to the left, right, output wires, namely $[\mathbf{L}] = [\mathbf{G}]\mathbf{a}, [\mathbf{R}] = [\mathbf{H}]\mathbf{b}, [\mathbf{O}] = [\mathbf{F}]\mathbf{c}$. Note that these commitments are of size $O(\text{poly}(\kappa)w)$ but independent of d. Next, it proves that $[\mathbf{O}], [\mathbf{L}], [\mathbf{R}]$

- lie in the image of $[\mathbf{M}_1]$ using the witness $\mathbf{c}$,
- satisfy the Hadamard relations: it computes a commitment $[\mathbf{Z}] = [(\mathbf{G}\otimes\mathbf{H})](\mathbf{a}\otimes\mathbf{b})$ and shows using the linear argument that the vector $\begin{bmatrix}\mathbf{Z}\\ \mathbf{O}\end{bmatrix}$ lies in the image of $\mathbf{M}_2$ using the witness $\mathbf{a}\otimes\mathbf{b}$.

The verifier checks that (1) the linear proofs verify and (2) that $e([\mathbf{L}], [\mathbf{R}]) = e([\mathbf{Z}], [1])$. It also does some additional input/output consistency check which we omit for now and describe next.

Now, let's see the core of the extraction argument. The inductive claim goes as follows: If we set $[\mathbf{F}]$ extractable for the i-th level, namely we the set $S_i = \{iw+1, \ldots, (i+1)w\}$, then -conditioned on an accepting proof- extracting the level i-th level wires corresponds to the correct values $[\mathbf{c}_i]$ w.r.t. the input $\mathbf{c}_0$. We will handle the base case later when we discuss input/output consistency. For the inductive step, assume the statement is true for i. We show that it is true for $i+1$. We proceed as follows:

1. We first set $\mathbf{G}, \mathbf{H}$ extractable at set S_{i+1} corresponding to the $i+1$-th level in addition to the $\mathbf{F}$ extractable at S_i. By the no-signaling guarantees the value $[\mathbf{c}_i]$ extracted by $[\mathbf{O}]$ is still correct.

2. By the local soundness of the linear quasi argument, the extracted values $[\mathbf{c}_i], [\mathbf{a}_{i+1}]$, $[\mathbf{b}_{i+1}]$ must lie in the image of the submatrix of $\mathbf{M}_1$ corresponding to these values. This matrix contains the blocks $\mathbf{I}, \mathbf{D}_{i+1}, \mathbf{E}_{i+1}$. Hence the values extracted correspond to the correct values $[\mathbf{a}_{i+1}], [\mathbf{b}_{i+1}]$ w.r.t. the input $\mathbf{c}_0$.

3. We only set $\mathbf{G}, \mathbf{H}$ extractable at set S_{i+1} and leave $\mathbf{F}$ extractable at the empty set. By the no-signaling guarantees the extracted wires for left and right values $[\mathbf{a}_{i+1}], [\mathbf{b}_{i+1}]$ are still correct.

4. In addition to $\mathbf{G}, \mathbf{H}$ extractable at set S_{i+1}, we set $\mathbf{F}$ extractable at S_{i+1}. Now we argue about local constraint of the Hadamard product. We proceed in two steps:
   - By the pairing test $e([\mathbf{L}], [\mathbf{R}]) = e([\mathbf{Z}], [1])$ and the assumption that $[\mathbf{a}_{i+1}]$, $[\mathbf{b}_{i+1}]$ are correct we get that

     $$\mathbf{T_G L} \otimes \mathbf{T_H R} = (\mathbf{T_G} \otimes \mathbf{T_H})(\mathbf{L} \otimes \mathbf{R}) = (\mathbf{T_G} \otimes \mathbf{T_H})\mathbf{Z} = \mathbf{T_{G \otimes H} Z}$$

     which implies that $\mathbf{z}_{i+1} = \mathbf{a}_{i+1} \otimes \mathbf{b}_{i+1}$. This means that the extracted value of the Kronecker commitment corresponds to the Kronecker product $\mathbf{a}_{i+1} \otimes \mathbf{b}_{i+1}$ of left and right wires in level $i+1$.
   - Working similarly to the step (2), we get that the extracted values $\mathbf{Z}_{i+1}, \mathbf{O}_{i+1}$ live in the image of $\mathbf{M}_2$. It should then be the case that we extract $[\mathbf{c}_{i+1}]$ which is the Hadamard product $\mathbf{a}_{i+1} \circ \mathbf{b}_{i+1}$. This correspond to the correct assignment of output wires in level $i+1$.

5. Finally, we only set $\mathbf{F}$ extractable at set S_{i+1} and leave $\mathbf{G}, \mathbf{H}$ extractable at the empty set. By the no-signaling guarantees the extracted value $[\mathbf{c}_{i+1}]$ is still correct.

We note that proving this is technically more involved. We need to show that the quasi arguments can be composed well, and they still satisfy the no-signaling properties despite the fact that they share commitment keys. Equivalently one could define and analyze a unified quasi argument to directly work with the circuit "transition funciton". In any case, we omit these details from these technical overview.

Input/Output Consistency. We modify the commitment $\mathbf{F}$ by making it trivially extractable at the input/output levels $0, d$ always, regardless of the extraction set. That is, we "use"

the identity matrix $\mathbf{I}_w$ for committing to the output wires at the first and last level. This corresponds to augmenting $\mathbf{F}$ with some identity rows. Thus, the verifier can always trivially check the consistency with input/output. Note that the final commitment size grows by $2|w|$, the size of input and output, but these values are part of the statement and don't need to be included in the proof. We stress out the "trivial" identity commitment satisfies the properties needed to be used in our quasi-arguments.

Assumptions. We next discuss the assumptions we use. For the specific matrices used in the reduction, one can prove soundness of the QA-NIZK argument under falsifiable assumptions since the S-submatrices $\mathbf{M}_1, \mathbf{M}_2$ produce a non-trivial subspace. This means that we rely on the kernel assumption we use for instantiating the QA-NIZK. Noting that MDDH assumptions implies the corresponding kernel assumptions, we can instantiate the quasi argument using the DLIN assumption. Furthermore, the no-signaling property of the commitment keys (the only computational property we use) reduces to an MDDH which we chose on instantiation. Noting that DDH does not hold in symmetric groups we resort to the DLIN assumption which makes the commitments larger by 1 group element. Thus, soundness of the above delegation scheme reduces to the DLIN assumption.

4.1.3.2 Overcoming the dependence on space/width.

The issue with the above construction is that setting $K = \mathcal{O}(|\mathtt{st}|)$ yields a proof whose size is linear in the space of the computation. To achieve succinctness in the general case, we need to also perform some "compressing" of the state/width. Kalai et. al. overcome this by considering delegation of RAM computation [KP16] using collision-resistant hash function to compress the width. They use a notion similar to the knowledge transfer notion, namely that no PPT adversary can produce digests $\mathtt{h}, \mathtt{h}'$ and state $\mathtt{st}$ such that $\mathtt{h} = \mathsf{Hash}(\mathtt{st})$ but $\mathtt{h}' \neq \mathsf{Hash}(f(\mathtt{st}))$. Now, a quasi argument for the local constraints $\mathtt{h}_i = \mathsf{Hash}(f(\mathtt{st}_i))$ and $\mathtt{h}_{i+1} = \mathsf{Hash}(f(\mathtt{st}_i))$ is enough for delegation in the general case.

While previous works achieve this by essentially encoding the computation of generic hash functions in the computation, we use hash functions that are based on Pedersen commitments and have nice algebraic structure and properties. This allows to avoid the concrete cost of encoding arbitrary hash functions in the arithmetic circuit. To this end, we use techniques from [GR19] to derive a structure preserving construction. We present next the basic ideas of their (low depth) delegation construction.

Structure Preserving Delegation for Low-Depth Circuits. González and Ráfols [GR19] constructed a delegation scheme with proof-size $\mathcal{O}(d\kappa)$ and verification requiring n plus $\mathcal{O}(d)$ cryptographic operations, where n is the size of the input, d the depth of the circuit and κ a security parameter. Interestingly, the verification procedure of [GR19] can be described completely as a set of pairing product equations. As shown by Abe et. al. [AFG+16], cryptographic primitives whose correctness can be stated as equations over bilinear groups are more suited for practically efficient arguments without resorting to generic reductions to a circuit or a 3CNF formula.

In the heart of the delegation scheme of [GR19] lie the two knowledge transfer arguments for linear and quadratic relations described before. To delegate the computation of an arithmetic circuit, the multiplication gates are partitioned in d levels. Each level groups the gates at the same distance from the inputs, without counting linear gates. In this way, the inputs of level $i+1$ are linear combinations of outputs of the i previous levels. A prover commits to the left, right, and output wires of each level as L_i, R_i, O_i. In the first d arguments f is a linear function and the argument handles the linear relations between the input wires (the openings of L_i, R_i) of level i and the output wires of all previous levels (the openings of $O_1, \ldots, O_{i-1}$). In the next d arguments f is the Hadamard product so that the opening of O_i is the Hadamard product of the openings of L_i and R_i. The fact that the verifier can check the commitment to the first level using the public input and a simple inductive argument over the levels shows that the output must be correct.

More concretely, starting from a correct commitment O_0 (directly checked for consistency with input x from the verifier) we conclude that L_1, R_1 by the knowledge transfer guarantee of the linear argument. Since L_1, R_1 are correct w.r.t. x, O_1 is also correct w.r.t. x by the knowledge transfer guarantee of the quadratic argument. We continue this way and we conclude that O_d is a correct commitment to the output of the computation. Now, we simply need to check that the claimed output y is a correct opening for that latter commitment.

As for soundness, the quadratic knowledge transfer argument requires a specific (not uniform) distribution for the commitment keys where each row of the matrix of the commitment key is the result of evaluating Lagrange polynomials at a different random point. Thus, soundness relies on a width-size assumption, namely "$\mathcal{R}$-Rational Strong Diffie Hellman" assumption [GR19] which is proven secure in the Generic Group Model. We stress out that we modify the construction of [GR19] to overcome the need for a q-size assumption and rely only on a constant-size one, albeit at the cost of having a quadratic srs and prover computation.

Succinct Publicly Verifiable Delegation for polynomial size circuits. We use the technique of [GR19] to overcome the width dependency in the above construction. The problem with this construction is that we need to rely on simple soundness of the underlying Kiltz and Wee QA-NIZK. However if we try to "shrink" the per-level information to eliminate the width dependence, the subspaces used become trivial and knowledge soundness seems to be needed.

We overcome this by relying on the knowledge transfer analysis of Kiltz and Wee used in [GR19]. To exploit this to construct delegation, we proceed as follows: we keep the same skeleton of the small-width circuit protocol, but instead of directly committing to the left, right and output wires, we commit to commitments of them. That is, for each level we compute three shrinking commitments -with size independent of the width- corresponding to left, right and output wires for that level, and we commit to these commitments (by including appropriate group elements in the srs). Furthermore, we use the knowledge transfer variants of the quasi arguments.

Now, our no-signaling extractor works as in the small-width case, but instead of the wires for some level, it outputs the commitments for the wires in this level. By the knowledge transfer

guarantees, we establish that the extracted values for each level satisfy:

1. *if* O_i is a commitment to $\mathbf{c}_i$ *then* L_{i+1} and R_{i+1} are commitments to $\mathbf{a}_{i+1}, \mathbf{b}_{i+1}$,
2. *if* L_{i+1} and R_{i+1} are commitments to $\mathbf{a}_{i+1}$ and $\mathbf{b}_{i+1}$ respectively, *then* O_{i+1} is a commitment to $\mathbf{c}_{i+1}$

Extracting these values in a no-signaling way, as in the bounded space case, yields soundness for the delegation scheme. The analysis is almost the same and the only difference is that the knowledge transfer guarantee implies some hardness assumption (MDDH) on the distribution of matrices used as parameters, in this case, the width commitment keys. To satisfy this using constant size assumptions, we use a simple variation of Pedersen commitments where the commitment keys satisfy the DLIN assumption.

Remark (Uniform vs Non-Uniform Computation)**.** Our construction can be used for any non-uniform computation, namely polynomial size arithmetic circuits, while previous works such as [PR17; KPY19] focus on delegating uniform computations: Turing or RAM machines. While this is a stronger result, we achieve it using a long (quadratic in the size/time of computation) srs, while the work of [KPY19] achieves a short (i.e. sublinear) srs. One motivation for working directly with poly-size circuits is for practical efficiency: we utilize the rich SNARK toolbox without the need to encode expensive cryptographic operations as arithmetic circuits, namely, we focus on structure preserving constructions. While we have an inefficient (quadratic) prover, in all other aspects we achieve optimal efficiency comparable with SNARKs from non-falsifiable assumptions. We believe that this is a promising direction and an interesting open problem is to improve the prover to quasi-linear using these techniques. This would yield a delegation scheme for poly-size circuits that directly competes with the aforementioned non-falsifiable based constructions in all aspects, effectively eliminating the need of using non-falsifiable assumptions in the context of deterministic computation. We also leave as future work exploring to what extend our techniques can be applied for delegating uniform computations and if this would give some improvement over existing constructions.

Remark (On bootstrapping and proof composition)**.** To improve efficiency (reduce srs size), [KPY19] use the bootstrapping technique which involves proof composition. Our techniques seem to be incompatible with the bootstrapping technique. This is because the srs of our construction depends on the circuit and we cannot directly reuse an srs for different computations. We leave as future work to examine if we can modify our techniques to be able to apply the bootstrapping technique. We also stress out that this might prove to be an interesting direction for improvements in practical efficiency as well due to some recent results in proof-composition techniques [BCMS20; BCL+21].

4.1.4 NIZK, SNARKs and Compact NIZK

We can use standard techniques to turn our delegation scheme into a NIZK argument. Essentially, the prover needs to prove knowledge of (additional) secret input wires w and proof

that $C(x, w) = y$ for some secret input w. Given the "structure preserving" properties of our delegation scheme, we can directly apply the Groth Sahai proof system [GS08][10] on the set of verification equations. In general, all we need to achieve knowledge soundness is an extractable (and hiding) commitment for extracting the witness w. Depending on the properties of the extractable commitment scheme we get different NIZK flavors.

If the commitments to the inputs are succinct, the construction yields a SNARK for **NP**. Such commitments are widely employed in SNARKs, but their security relies on non-standard assumptions: either knowledge type assumptions such as q-Knowledge of Exponents assumption [GGPR13] or the generic group model [Gro16]. If we take for example the zk-SNARK from [DFGK14], the size of q is the number of field elements extracted from a valid proof. Indeed, the proof of soundness requires the extraction of all the circuit wires, which are later used to break some falsifiable q-assumption. Consequently, the knowledge assumption is of size $q = O(|C|)$. By reducing the number of extracted values from $O(|C|)$ to $|w|$, we reduce the size of the underlying knowledge assumption to $q = |w| < |C|$.

If we use the "bit-string" argument of [GHR15] to show knowledge of $\mathbf{b} \in \{0,1\}^n$, we get extractable commitments of size $n + O(1)$ group elements based on a constant-size falsifiable assumption. Combining this extractable commitment with our delegation scheme yields a NIZK argument for circuit satisfiability with proof size $n + O(1)$ groups elements, or equivalently of size $O(n\kappa)$.

Finally, we can then use the techniques of Katsumata et. al. [KNYY19; KNYY20] to construct a compact NIZK. The construction of Katsumata et al. is based on a non-compact NIZK argument for NC^1 plus a symmetric key encryption scheme (KeyGen, Enc, Dec) where the size of $\mathsf{Enc}_K(m)$ is $|m| + \mathsf{poly}(\kappa)$. Instead of committing to the input $\mathbf{x}$ of a circuit C, we need to compute $K \leftarrow \mathsf{KeyGen}(1^\kappa)$ to obtain $\mathsf{ct} \leftarrow \mathsf{Enc}(K, \mathbf{x})$ and give a NIZK argument of knowledge of some $K \in \{0,1\}^{\mathsf{poly}(\kappa)}$ such that $C(\mathsf{Dec}(K, \mathsf{ct})) = 1$. We note that we can straightforward use this idea to construct compact NIZK for any circuit by simply plugging our NIZK argument based on the commitments of [GHR15]. The final proof is of size $|\mathsf{ct}| + |K|\,\mathsf{poly}(\kappa) + |\pi| = n + \mathsf{poly}(\kappa)$ and is sound for any polynomial size circuit.

4.2 Knowledge Transfer Arguments

In this section we recall arguments of knowledge transfer for membership in linear spaces as defined in [GR19] which in turn is an instantiation of [KW15] with a different security analysis. We also slightly modify the construction to turn it into an argument of knowledge transfer for the sum language, which we will use in later constructions.

Let gk be a bilinear group of order p and $\mathcal{N}, \mathcal{N}, \mathcal{P}, \mathcal{Q}$ be matrix distributions outputting

[10] This can be also achieved in a more efficient way (concretely) by directly using hiding commitments for the delegation scheme.

Figure 4.1 Construction KTLin for Lin-$\mathcal{L}^{\text{yes}}$, Lin-$\mathcal{L}^{\text{no}}$. For $\ell_1 = \ell_2$, construction KTSum for Sum-$\mathcal{L}^{\text{yes}}$, Sum-$\mathcal{L}^{\text{no}}$ is identical with the only difference that $\mathbf{K}_2 = \mathbf{K}_1$.

K(gk, $[\mathbf{M}]_1, [\mathbf{N}]_2, [\mathbf{P}]_1, [\mathbf{Q}]_2$):

$\mathbf{K}_1 \leftarrow \mathbb{F}^{\ell_1 \times \bar{k}}; \mathbf{K}_2 \leftarrow \mathbb{F}^{\ell_2 \times \bar{k}}; \mathbf{K}_3 \leftarrow \mathbb{F}^{\ell_3 \times \bar{k}}; \mathbf{K}_4 \leftarrow \mathbb{F}^{\ell_4 \times \bar{k}}$

Sample $\mathbf{A} \leftarrow \mathcal{D}_k; \mathbf{\Gamma} \leftarrow \mathbb{F}^{n \times \bar{k}}$

$[\mathbf{B}]_1 = [\mathbf{M}^\top \mathbf{K}_1 + \mathbf{N}^\top \mathbf{K}_2 + \mathbf{\Gamma}]_1; [\mathbf{D}]_2 = [\mathbf{P}^\top \mathbf{K}_3 + \mathbf{Q}^\top \mathbf{K}_4 - \mathbf{\Gamma}]_2$

$\mathbf{C}_1 = \mathbf{K}_1 \mathbf{A}; \mathbf{C}_2 = \mathbf{K}_2 \mathbf{A}; \mathbf{C}_3 = \mathbf{K}_3 \mathbf{A}; \mathbf{C}_4 = \mathbf{K}_4 \mathbf{A}$

Output srs = $(gk, [\mathbf{A}]_{1,2}, [\mathbf{B}]_1, [\mathbf{D}]_2, [\mathbf{C}_1]_2, [\mathbf{C}_2]_1, [\mathbf{C}_3]_2, [\mathbf{C}_4]_1)$

Prove(srs, $([\mathbf{c}_1]_1, [\mathbf{c}_2]_2, [\mathbf{d}_1]_1, [\mathbf{d}_2]_2)$, w):

Sample $\rho \leftarrow \mathbb{F}^{\bar{k}}; [\pi]_1 = \mathsf{w}^\top [\mathbf{B}]_1 + [\rho]_1; [\theta]_2 = \mathsf{w}^\top [\mathbf{D}]_2 - [\rho]_2$

Output $([\pi]_1, [\theta]_2)$

Verify(srs, $([\mathbf{c}_1]_1, [\mathbf{c}_2]_2, [\mathbf{d}_1]_1, [\mathbf{d}_2]_2), ([\pi]_1, [\theta]_2)$):

Output 1 iff

$$e([\pi]_1, [\mathbf{A}]_2) + e([\theta]_2, [\mathbf{A}]_1) = e([\mathbf{c}_1^\top]_1, [\mathbf{C}_1]_2) - e([\mathbf{c}_2^\top]_2, [\mathbf{C}_2]_1) - e([\mathbf{d}_1^\top]_1, [\mathbf{C}_3]_2) - e([\mathbf{d}_2^\top]_2, [\mathbf{C}_4]_1)$$

matrices $[\mathbf{M}]_1 \in \mathbb{G}_1^{\ell_1 \times n}, [\mathbf{N}]_2 \in \mathbb{G}_2^{\ell_2 \times n} [\mathbf{P}]_1 \in \mathbb{G}_1^{\ell_3 \times n} [\mathbf{Q}]_2 \in \mathbb{G}_2^{\ell_4 \times n}$ respectively. In Fig. 4.1, we present two arguments of knowledge transfer for (1) the linear membership language

$$\text{Lin-}\mathcal{L}^{\text{yes}} = \left\{ ([\mathbf{c}_1]_1, [\mathbf{c}_2]_2, [\mathbf{d}_1]_1, [\mathbf{d}_2]_2) \mid \exists \mathbf{w} \text{ s.t } \begin{pmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{M} \\ \mathbf{N} \end{pmatrix} \mathbf{w} \text{ and } \begin{pmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{P} \\ \mathbf{Q} \end{pmatrix} \mathbf{w} \right\}$$

$$\text{Lin-}\mathcal{L}^{\text{no}} = \left\{ ([\mathbf{c}_1]_1, [\mathbf{c}_2]_2, [\mathbf{d}_1]_1, [\mathbf{d}_2]_2, \mathbf{w}) \mid \begin{pmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{M} \\ \mathbf{N} \end{pmatrix} \mathbf{w} \text{ and } \begin{pmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \end{pmatrix} \neq \begin{pmatrix} \mathbf{P} \\ \mathbf{Q} \end{pmatrix} \mathbf{w} \right\}$$

and (2) the sum knowledge transfer language

$$\text{Sum-}\mathcal{L}^{\text{yes}} = \left\{ ([\mathbf{c}_1]_1, [\mathbf{c}_2]_2, [\mathbf{d}_1]_1, [\mathbf{d}_2]_2) \mid \exists \mathbf{w} \text{ s.t } \mathbf{c}_1 + \mathbf{c}_2 = (\mathbf{M} + \mathbf{N})\mathbf{w} \text{ and } \begin{pmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{P} \\ \mathbf{Q} \end{pmatrix} \mathbf{w} \right\}$$

$$\text{Sum-}\mathcal{L}^{\text{no}} = \left\{ ([\mathbf{c}_1]_1, [\mathbf{c}_2]_2, [\mathbf{d}_1]_1, [\mathbf{d}_2]_2, \mathbf{w}) \mid \mathbf{c}_1 + \mathbf{c}_2 = (\mathbf{M} + \mathbf{N})\mathbf{w} \text{ and } \begin{pmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \end{pmatrix} \neq \begin{pmatrix} \mathbf{P} \\ \mathbf{Q} \end{pmatrix} \mathbf{w} \right\}$$

A knowledge transfer argument is just an argument for the promise problem defined by $\mathcal{L}^{\text{yes}}$ and $\mathcal{L}^{\text{no}}$. Completeness means that an honest proof is accepting for any statement in $\mathcal{L}^{\text{yes}}$. Soundness that any proof for a statement in $\mathcal{L}^{\text{no}}$, which comes with an "advice" $\mathbf{w}$, is accepting only with negligible probability.

We use this construction with (1) $\mathbf{N} = \mathbf{0}$ for the case of linear knowledge transfer and (2) $\mathbf{Q} = \mathbf{0}$ for the case of sum knowledge transfer so we consider proofs for these two cases. We stress out that the proofs are easily extended to accommodate for the more general cases.

For the case of KTLin, when setting $\mathbf{N} = \mathbf{0}$, the security is shown in [GR19]. Essentially, they show that the construction is sound if the image of the matrices outputted by $\mathcal{M}$ is pseudorandom. Since this condition holds under the $\mathcal{M}$-MDDH, the construction is sound for these parameters. Noting that this condition also holds for the case of distributions that satisfy the KMDDH assumption (Thm. 1), these distributions can also be used to select language parameters. The extension for Sum-$\mathcal{L}$ is a direct adaptation of the techniques of [GR19].

We summarize the results in the following theorem.

Theorem 9. *Let $\mathcal{D}_k$ be a matrix distribution that satisfies the $\mathcal{D}_k$-*SKerMDH *assumption. Then*

1. *If $\mathcal{M}^\top$-*MDDH *holds, then construction* KTLin *is sound w.r.t. language* Lin-$\mathcal{L}^{\text{no}}$.
2. *If $(\mathcal{U} \otimes \mathcal{V})^\top$-*KMDDH *holds and $\alpha = ([\mathbf{M}]_1, [\mathbf{N}]_2, [\mathbf{U}]_1, [\mathbf{V}]_2)$ where $\alpha \leftarrow (\mathcal{U} \otimes \mathcal{V})$, then construction* KTSum *is sound w.r.t. language* Sum-$\mathcal{L}^{\text{no}}$ *against adversaries that know α.*

Remark 1. If the construction is sound for language parameters $\mathcal{M}, \mathcal{N}, \mathcal{P}, \mathcal{Q}$, then it is also sound when the language is defined by distributions $\mathcal{M}, \mathcal{N}, \mathcal{P}, \mathcal{Q}$ augmented with additional zero columns. This is proven in [GR19, Lemma 15]. Essentially one can reduce to the knowledge transfer argument where we delete the zero columns of the matrix and rely on the linearity properties of the proofs of the construction.

4.3 No-Signaling Somewhere Statistically Binding Commitment Schemes

In this section we recall Somewhere Statistically Binding (SSB) commitments and then define two additional notions for them: no-signaling extraction and oblivious key generation. The former is a natural adaptation of the definitions of no-signaling extractors from previous works [PR17; KPY19]. We show that the latter implies the former, and we give an efficient instantiation based on any $\mathcal{D}_k$-MDDH assumption. Finally, we consider the Kronecker product of two of these commitments.

An SSB commitment scheme, as the name suggests, is statistically binding only w.r.t. some variables which are determined during key generation. The commitment key computationally hides any information about this set, meaning that for all "modes" the commitment keys are computationally indistinguishable. Furthermore, the KeyGen algorithm outputs a trapdoor which allows to extract (a function of) the values in this set.

The definition that follows only considers extracting a subset $S \subseteq [n]$ of the coordinates with the only restriction that its size is bounded by the locality parameter K. We emphasize, however, that one can generalize and consider extracting any set from a subset family $\mathcal{S} \subseteq 2^{[n]}$. We will in fact consider this in one of our construction but for the sake of simplicity we choose to present

Definition 24. Let $[\cdot] : \mathcal{M} \to G$ be a function, where $\mathcal{M}$ is the message space and G some set. Syntactically, a Somewhere Statistically Binding Commitment Scheme CS is a tuple of algorithms CS = (KeyGen, Com, Extract) where

- $(\mathsf{ck}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{gk}, \mathbf{n}, K, S)$: KeyGen takes as input the parameters gk, $n \in \mathbb{N}$, locality parameter $K \in [n]$ and the set $S \subseteq [n], |S| \leq K$. It outputs a commitment key ck, which may also contain some auxiliary information aux, and a secret key sk, containing a trapdoor τ and possibly the random coins used by KeyGen.
- $\mathsf{c} \leftarrow \mathsf{Com}(\mathsf{ck}, \mathbf{x})$: Com takes as input the commitment key ck and a vector $\mathbf{x} \in \mathcal{M}^n$ and outputs a commitment c,
- $\mathbf{y} \leftarrow \mathsf{Extract}(\tau, \mathsf{c})$: Extract takes as input the trapdoor τ and a commitment c, and outputs the value $\mathbf{y} \in G$ allegedly equaling $[\mathbf{x}_S]$, where $\mathbf{x}$ is a valid opening for c.

For all $\kappa \in \mathbb{N}, n \in \mathbb{N}, K \in [n], S_0, S_1 \subseteq [n]$ with $|S_0|, |S_1| \leq \mathbf{K}$, CS must satisfy the following properties:

- Index Set Hiding: for all PPT $\mathcal{D}$

$$\Pr_{\mathsf{gk} \leftarrow \mathcal{G}(1^\kappa)} \left[\mathcal{D}(\mathsf{ck}) = b \;\middle|\; \begin{array}{r} b \leftarrow \{0,1\} \\ (\mathsf{ck}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{gk}, n, K, S_b) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\kappa)$$

- Somewhere Statistically Binding: for all all, even unbounded $\mathcal{A}$,

$$\Pr_{\mathsf{gk} \leftarrow \mathcal{G}(1^\kappa)} \left[\begin{array}{c} \mathsf{Com}(\mathsf{ck}, \mathbf{x}) = \mathsf{Com}(\mathsf{ck}, \mathbf{x}') \\ \text{and } \mathbf{x}_S \neq \mathbf{x}'_S \end{array} \;\middle|\; \begin{array}{r} (\mathsf{ck}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{gk}, n, K, S) \\ (\mathbf{x}, \mathbf{x}') \leftarrow \mathcal{A}(\mathsf{ck}) \end{array} \right] \leq \mathsf{negl}(\kappa)$$

- G-Extractability: for all, even unbounded $\mathcal{A}$

$$\Pr_{\mathsf{gk} \leftarrow \mathcal{G}(1^\kappa)} \left[\begin{array}{c} \exists \mathbf{x} \text{ s.t. } \mathsf{c} = \mathsf{Com}(\mathsf{ck}, \mathbf{x}) \\ \text{and } \mathbf{y} \neq [\mathbf{x}_S] \end{array} \;\middle|\; \begin{array}{r} (\mathsf{ck}, \mathsf{sk} = (\tau, r)) \leftarrow \mathsf{KeyGen}(\mathsf{gk}, n, K, S) \\ \mathsf{c} \leftarrow \mathcal{A}(\mathsf{pk}) \\ \mathbf{y} \leftarrow \mathsf{Extract}(\tau, \mathsf{c}) \end{array} \right] \leq \mathsf{negl}(\kappa)$$

Note that an SSB commitment is also "everywhere" computationally binding. This is the case since a breach in binding, namely the ability to produce c that opens to both $\mathbf{x} \neq \mathbf{x}'$, implies the ability to distinguish where the commitment is not statistically binding contradicting the index set hiding property.

We next present an extra property for an SSB commitment scheme which we call no-signaling extraction and is a natural adaptation of the definitions of [PR17; KPY19].

Definition 25. We say the extractor of an SSB commitment scheme CS = (KeyGen, Com, Extract) with commitment space C[11] no-signaling if for any $S' \subseteq S \subseteq [n]$, where $|S| \leq K$,

[11] We assume that membership in C is efficiently decidable

and any PPT adversary $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2)$,

$$\left| \Pr_{\mathsf{gk} \leftarrow \mathcal{G}(1^\kappa)} \left[\mathcal{D}_2(\mathsf{ck}_{S'}), \mathsf{c}, \mathbf{y}') = 1 \;\middle|\; \begin{array}{r} (\mathsf{ck}_{S'}, \mathsf{sk}_{S'}) \leftarrow \mathsf{KeyGen}(\mathsf{gk}, n, K, S') \\ \mathsf{c} \leftarrow \mathcal{D}_1(\mathsf{ck}_{S'}) \\ \text{if } \mathsf{c} \notin \mathcal{C}\text{: } \mathsf{c} \leftarrow \perp \\ \mathbf{y}' \leftarrow \mathsf{Extract}(\tau, \mathsf{c}), \text{ where } \mathsf{sk}_{S'} = (\tau, r) \end{array} \right] - \right.$$
$$\left. \Pr_{\mathsf{gk} \leftarrow \mathcal{G}(1^\kappa)} \left[\mathcal{D}_2(\mathsf{ck}_S, \mathsf{c}, \mathbf{y}_{S'}) = 1 \;\middle|\; \begin{array}{r} (\mathsf{ck}_S, \mathsf{sk}_S) \leftarrow \mathsf{KeyGen}(\mathsf{gk}, n, K, S) \\ \mathsf{c} \leftarrow \mathcal{D}_1(\mathsf{ck}_S) \\ \text{if } \mathsf{c} \notin \mathcal{C}\text{: } \mathsf{c} \leftarrow \perp \\ \mathbf{y} \leftarrow \mathsf{Extract}(\tau, \mathsf{c}), \text{ where } \mathsf{sk}_S = (\tau, r) \end{array} \right] \right| \leq \mathsf{negl}(\kappa)$$

We define also oblivious trapdoor generation. This property states that there exists an oblivious key generation algorithm, that takes a commitment key ck that allows extraction in S and a set $S' \subseteq S$, and can produce a fresh commitment key ck′ and a trapdoor to extract S'. The distribution of the new key ck′ is statistically close to that of ck and -importantly- the oblivious key generation algorithm does not get as input the original extraction set S. In other words, given a commitment key ck that we know allows extraction for some superset of S', we can create a new key *with* a trapdoor for S' without skewing the distribution of ck.

Definition 26. An SSB commitment scheme has oblivious trapdoor generation if there exists a PPT algorithm OblKeyGen such that for all $\kappa \in \mathbb{N}, n \in \mathbb{N}, K \in [n]$, $S \subseteq [n]$, with $|S| \leq K$, $S' \subseteq S$, and for all, even unbounded $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2)$,

$$\left| \Pr_{\mathsf{gk} \leftarrow \mathcal{G}(1^\kappa)} \left[\mathcal{D}_2(\mathsf{ck}', \mathsf{c}, \mathbf{y}') = 1 \;\middle|\; \begin{array}{r} (\mathsf{ck}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{gk}, n, K, S) \\ (\mathsf{ck}', \tau') \leftarrow \mathsf{OblKeyGen}(\mathsf{gk}, n, K, S', \mathsf{ck}) \\ \mathsf{c} \leftarrow \mathcal{D}_1(\mathsf{ck}') \\ \mathbf{y}' \leftarrow \mathsf{Extract}(\tau', c), \text{ where } \mathsf{sk} = (\tau, r) \end{array} \right] - \right.$$
$$\left. \Pr_{\mathsf{gk} \leftarrow \mathcal{G}(1^\kappa)} \left[\mathcal{D}_2(\mathsf{ck}, \mathsf{c}, \mathbf{y}_{S'}) = 1 \;\middle|\; \begin{array}{r} (\mathsf{ck}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{gk}, n, K, S) \\ c \leftarrow \mathcal{D}_1(\mathsf{ck}) \\ \mathbf{y} \leftarrow \mathsf{Extract}(\tau, \mathsf{c}), \text{ where } \mathsf{sk} = (\tau, r) \end{array} \right] \right| \leq \mathsf{negl}(\kappa)$$

Next, we show that an SSB commitment scheme with oblivious trapdoor generation is also no-signaling.

Theorem 10. *Let* $\mathsf{CS} = (\mathsf{KeyGen}, \mathsf{OblKeyGen}, \mathsf{Com}, \mathsf{Extract})$ *be an SSB commitment scheme with oblivious trapdoor generation and ISH. Then,* CS *is also no-signaling.*

Proof. Fix any $S' \subseteq S \subseteq [n]$ with $|S| \leq K$, and let $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2)$ be a distinguisher against no signaling extraction for these values. We show by a sequence of games that its success probability is negligible.

$\mathsf{Game}_0^{\mathcal{D}}(1^\kappa)$: In this game, we execute $(\mathsf{ck}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{gk}, n, K, S)$. We then get $\mathsf{c} \leftarrow \mathcal{D}_1(\mathsf{ck})$, change it to $\perp$ if $\mathsf{c} \notin \mathcal{C}$, and compute $\mathbf{y} \leftarrow \mathsf{Extract}(\tau, \mathsf{c})$ for $\mathsf{sk} = (\tau, r)$. The output is $\mathcal{D}_2(\mathsf{ck}, \mathsf{c}, \mathbf{y}_{S'})$.

$\mathsf{Game}_1^{\mathcal{D}}(1^\kappa)$: In this game, we execute $(\mathsf{ck}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{gk}, n, K, S)$ and $(\mathsf{ck}_{\mathsf{obl}}, \tau_{\mathsf{obl}}) \leftarrow \mathsf{OblKeyGen}(\mathsf{gk}, n, K, S', \mathsf{ck})$. We then get $\mathsf{c} \leftarrow \mathcal{D}_1(\mathsf{ck}_{\mathsf{obl}})$, change it to $\perp$ if $\mathsf{c} \notin C$, and compute $\mathbf{y}' \leftarrow \mathsf{Extract}(\tau_{\mathsf{obl}}, \mathsf{c})$. The output is $\mathcal{D}_2(\mathsf{ck}_{\mathsf{obl}}, \mathsf{c}, \mathbf{y}')$.

$\mathsf{Game}_2^{\mathcal{D}}(1^\kappa)$: In this game, we execute $(\mathsf{ck}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{gk}, n, K, S')$ and $(\mathsf{ck}_{\mathsf{obl}}, \tau_{\mathsf{obl}}) \leftarrow \mathsf{OblKeyGen}(\mathsf{gk}, n, K, S', \mathsf{ck})$. We then get $\mathsf{c} \leftarrow \mathcal{D}_1(\mathsf{ck}_{\mathsf{obl}})$, change it to $\perp$ if $\mathsf{c} \notin C$, and compute $\mathbf{y}' \leftarrow \mathsf{Extract}(\tau_{\mathsf{obl}}, \mathsf{c})$. The output is $\mathcal{D}_2(\mathsf{ck}_{\mathsf{obl}}, \mathsf{c}, \mathbf{y}')$.

$\mathsf{Game}_3^{\mathcal{D}}(1^\kappa)$: In this game, we execute $(\mathsf{ck}', \mathsf{sk}') \leftarrow \mathsf{KeyGen}(\mathsf{gk}, n, K, S')$. We then get $c \leftarrow \mathcal{D}_1(\mathsf{ck}')$, change it to $\perp$ if $\mathsf{c} \notin C$, and compute $\mathbf{y} \leftarrow \mathsf{Extract}(\tau', \mathsf{c})$ for $\mathsf{sk} = (\tau, r)$. The output is $\mathcal{D}_2(\mathsf{ck}, \mathsf{c}, \mathbf{y}')$.

Now we show the output of games i and $i+1$ is indistinguishable for $i = 0$ to 2.

- *Cases* $i = 0$, $i = 2$. For $i = 0$, the two games are distributed identically to the two cases of the oblivious trapdoor generation definition for $S' \subseteq S$. Thus, the outputs of the games are statistically close. For $i = 2$, the same argument holds for $S = S'$. Note that in both cases, the oblivious trapdoor generation distinguisher is unbounded so it can compute $\mathsf{sk}_{\mathsf{obl}}$.

- *Case* $i = 1$. The difference in the two games is how we sample the $(\mathsf{ck}, \mathsf{sk})$ pair, either programmed to extract S or S'. By the index set hiding property the outputs of the two games are computationally indistinguishable.

Finally, noting that $\mathsf{Game}_0^{\mathcal{D}}$, $\mathsf{Game}_3^{\mathcal{D}}$ correspond to the two cases of no signaling extraction, the result follows. □

4.3.1 Algebraic SSB Commitments.

In this section, we define algebraic SSB commitments following the definition of algebraic commitment schemes of [RS20] and extend them to what we call *split* algebraic SSB commitments.

Informally, an algebraic SSB commitment scheme is a commitment scheme where the commitment key is a matrix $[\mathbf{G}]$ of group elements such that (1) committing to a vector $\mathbf{x}$ is done by multiplying on the left with $[\mathbf{G}]$, that is $[\mathbf{c}] = [\mathbf{G}]\mathbf{x}$ and (2) the trapdoor is a matrix of field elements $\mathbf{T}$ and local extraction is done by multiplying the commitment on the left with $\mathbf{T}^\top$, that is $[\mathbf{x}_S] = \mathbf{T}^\top[\mathbf{c}]$. We also allow the commitment key to output some public auxiliary information which is not used in committing nor extraction.

Definition 27. An SSB commitment scheme $\mathsf{CS} = (\mathsf{KeyGen}, \mathsf{Com}, \mathsf{Extract})$ is *algebraic* if, given $\mathsf{gk} \leftarrow \mathcal{G}(1^\kappa)$,

- $\mathsf{KeyGen}(\mathsf{gk}, n, K, S)$ outputs $\mathsf{pk} = [\mathbf{G}] \in \mathbb{G}^{\overline{K} \times n}$ and $\mathsf{sk} = (\mathbf{T} \in \mathbb{F}^{\overline{K} \times |S|}, \mathbf{G})$ where $\overline{K} \geq K$,

- $\mathsf{Com}([\mathbf{G}], \mathbf{x}) = [\mathbf{G}]\mathbf{x}$,
- $\mathbf{T}^\top \mathbf{G} = \mathbf{\Sigma}_\mathbf{S} \mathbf{P}_\mathbf{S}$.

We also define a subtype of algebraic commitments which are specific to asymmetric groups, where the commitment key is "split" between the two groups.

Definition 28. An SSB commitment scheme $\mathsf{CS} = (\mathsf{KeyGen}, \mathsf{Com}, \mathsf{Extract})$ is *split algebraic* if

- $\mathsf{KeyGen}(\mathsf{gk}, n, K, S)$ outputs $\mathsf{ck} = ([\mathbf{G}]_1 \in \mathbb{G}_1^{\overline{K} \times n}, [\mathbf{H}]_2 \in \mathbb{G}_2^{\overline{K} \times n})$ and $\mathsf{sk} = (\mathbf{T} \in \mathbb{F}^{\overline{K} \times |S|}, (\mathbf{G}, \mathbf{H}))$, for $\overline{K} \geq K$,
- $\mathsf{Com}([\mathbf{G}]_1, [\mathbf{H}]_2, \mathbf{x}) = ([\mathbf{G}]_1 \mathbf{x}, [\mathbf{H}]_2 \mathbf{x})$,
- $\mathbf{T}^\top \mathbf{G} + \mathbf{T}^\top \mathbf{H} = \mathbf{\Sigma}_S \mathbf{P}_\mathbf{S}$.

All SSB commitment schemes in this work are algebraic or split-algebraic. Note that all (split-)SSB commitments only differ on the key generation algorithm. For that reason we sometimes refer to a commitment key distribution as the commitment scheme itself.

In the case of non-split algebraic SSB commitments, we can $\mathbb{G}$-extract by computing

$$\mathbf{T}^\top [\mathbf{c}] = \mathbf{T}^\top [\mathbf{G}\mathbf{x}] = [\mathbf{\Sigma}_S \mathbf{P}_S \mathbf{x}] = [\mathbf{x}_S]$$

while in the case of split-algebraic commitments, we can only $\mathbb{G}_T$ extract. That is, we can compute values $[\mathbf{u}_S]_1, [\mathbf{v}_S]_2$ such that $e([\mathbf{u}_S]_1, [1]_2) + e([1]_1, [\mathbf{v}_S]_2) = [\mathbf{x}_S]_T$.

Indeed, if $[\mathbf{c}]_1 = [\mathbf{G}]_1 \mathbf{x}$ and $[\mathbf{d}]_2 = [\mathbf{H}]_2 \mathbf{x}$ then we can compute $[\mathbf{u}_S]_1 = \mathbf{T}[\mathbf{c}]_1$ and $[\mathbf{v}_S]_2 = \mathbf{T}[\mathbf{d}]_2$ and it holds that

$$\mathbf{u}_S + \mathbf{v}_S = \mathbf{T}^\top \mathbf{c} + \mathbf{T}^\top \mathbf{d} = \mathbf{T}^\top \mathbf{G}\mathbf{x} + \mathbf{T}^\top \mathbf{H}\mathbf{x} = (\mathbf{T}^\top \mathbf{G} + \mathbf{T}^\top \mathbf{H})\mathbf{x} = \mathbf{\Sigma} \mathbf{P}_S \mathbf{x} = \mathbf{x}_S$$

Note that by definition, if the commitment key generation does not fail, the commitments are perfectly binding/extractable at S. This will be the case for commitment schemes with perfect completeness. We will utilize this fact in our constructions to simplify some of the arguments.

4.3.2 Somewhere Statistically Binding Commitments with Oblivious Trapdoor Generation

We present in Fig. 4.2 a simple construction of an SSB with Oblivious Key Generation from plain SSB commitments with locality parameter 1. The setup algorithm instantiates K different commitment keys and, given a set S, each of the first $|S|$ commitment keys is extractable in a different position $s \in S$. The last $K - |S|$ are binding for the empty set. To commit to a

Figure 4.2 Oblivious SSB commitment scheme from K instantiations of SSB commitments with locality parameter 1.

$\mathsf{CS}'.\mathsf{KeyGen}(\mathsf{gk}, n, K, S)$**:**

For $s_i \in S$ set $(\mathsf{ck}_i, \tau_i) \leftarrow \mathsf{CS.KeyGen}(\mathsf{gk}, n, 1, \{s_i\})$

For $|S|+1 \leq i \leq K$ set $(\mathsf{ck}_i, \tau_i) \leftarrow \mathsf{CS.KeyGen}(\mathsf{gk}, n, 1, \emptyset)$

Set $\mathsf{ck} = (\mathsf{ck}_1, \ldots, \mathsf{ck}_K)$, $\tau = ((\tau_1, s_1), \ldots, (\tau_{|S|}, s_{|S|}))$ and output (ck, τ)

$\mathsf{CS}'.\mathsf{Com}(\mathsf{ck} = (\mathsf{ck}_1, \ldots, \mathsf{ck}_K), \mathbf{x})$**:**

For $1 \leq i \leq K$ compute $\mathsf{c}_i \leftarrow \mathsf{CS.Com}(\mathsf{ck}_i, \mathbf{x})$

Set $\mathsf{c} = (\mathsf{c}_1, \ldots, \mathsf{c}_K)$ and output c

$\mathsf{CS}'.\mathsf{Extract}(\tau = ((\tau_1, s_1), \ldots, (\tau_{|S|}, s_{|S|})), \mathsf{c} = (\mathsf{c}_1, \ldots, \mathsf{c}_K))$**:**

For all $s_i \in S$ compute $y_{s_i} \leftarrow \mathsf{CS.Extract}(\tau_i, \mathsf{c}_i)$

Set $\mathbf{y} = (y_1, \ldots, y_{|S|})$ and output $\mathbf{y}$

$\mathsf{CS}'.\mathsf{OblKeyGen}(\mathsf{gk}, n, K, S', a, \mathsf{ck} = (\mathsf{ck}_1, \ldots, \mathsf{ck}_K))$**:**

Parse a as $i_1, \ldots, i_{|S'|}$, the indices of the commitment keys binding at $s_{i_j} \in S'$

For $1 \leq j \leq |S'|$ set $(\mathsf{ck}_{i_j}, \tau_{i_j}) \leftarrow \mathsf{CS.KeyGen}(\mathsf{gk}, n, 1, \{s'_j\})$.

Set $\mathsf{ck} = (\mathsf{ck}_1, \ldots, \mathsf{ck}_K)$, $\tau = ((\tau_{i_1}, s_{i_1}), \ldots, (\tau_{i_{|S'|}}, s_{i_{|S'|}}))$ and output (ck, τ)

value $\mathbf{x}$, one gives K commitments to this value with each of the commitment keys. To verify an opening, one verifies each individual opening and that all the openings are the same.

Note that the ordering of the elements in S is arbitrary and, in some sense, there's no unique key generation algorithm for a set S. Indeed, it is only necessary that the commitment key contains K commitment keys for locality 1 such that $\mathsf{ck}_{i_1}, \ldots, \mathsf{ck}_{i_{|S|}}$ are binding at $s_1, \ldots, s_{|S|}$ respectively. Note that there are $\binom{K}{|S|}$ different choices of $i_1, \ldots, i_n$. For this reason, if the input of the oblivious generator is just S', it is impossible to know which commitment keys are the ones corresponding to S'. To alleviate this, the oblivious key generator receives as advice the indices where S' "appears" in S that is, $i_1, \ldots, i_{|S'|}$ such that $s_{i_1} = s'_1$.

In this case we need to change a little the proof that oblivious trapdoor generation implies no-signaling. We add a game $\mathsf{Game}^{\mathcal{D}}_{1/2}(1^\kappa)$, between games 0 and 1, which is identical to $\mathsf{Game}^{\mathcal{D}}_0(1^\kappa)$ but $\mathcal{E}_1$ samples ck_i binding at $\{s_i\}$ if $s_i \in S'$ and at $\emptyset$ if not. By the index-set hiding property of $\mathsf{ck}_1, \ldots, \mathsf{ck}_K$ the output of both games is indistinguishable. $\mathsf{Game}^{\mathcal{D}}_1(1^\kappa)$ is as before but the oblivious key generator receives also the advise. The rest of the proof is exactly as before

Theorem 11. *Let* CS *be an SSB commitment with locality parameter $K = 1$. Then construction* CS′ *of Fig. 4.2 is an SSB commitment with Oblivious Trapdoor Generation.*

Figure 4.3 MPed SSB commitment scheme with oblivious trapdoor generation parametrized by the matrix distribution $\mathcal{D}_k$.

KeyGen(gk, n, K, S)**:**

Let $\mathbf{A} \leftarrow \mathcal{D}_k$, $\mathbf{B} \leftarrow \mathbb{F}^{K+k\times K-|S|}$, $\mathbf{W} \leftarrow \mathbb{F}^{K-1\times k+1}$ and define $\mathbf{G}_0 = \begin{pmatrix} \mathbf{B} & \mathbf{A} \\ & \mathbf{WA} \end{pmatrix}$

Let $\mathbf{G}_S \leftarrow \mathbb{F}^{K+k\times|S|}$ and $\mathbf{\Gamma} \leftarrow \mathbb{F}^{K+k-|S|\times n-|S|}$

Let $\mathbf{T}_S \in \mathbb{F}^{K+k\times|S|}$ s.t. $\mathbf{T}_S^\top \mathbf{G}_S = \mathbf{I}_{|S|}$ and $\mathbf{T}_S^\top \mathbf{G}_0 = \mathbf{0}_{|S|\times K+k-|S|}$. Abort if such a matrix does not exist

Let $\mathbf{G} = (\mathbf{G}_S|\mathbf{G}_0\mathbf{\Gamma})\,\mathbf{P}_S$. Output $(\mathsf{ck}, \mathsf{sk}) = ([\mathbf{G}], (\mathbf{T}_S, \mathbf{G}))$

OblKeyGen(gk, n, K, S', ck $= [\mathbf{G}]_\mu$)**:** *// $S' \subseteq S$*:

Sample $\mathbf{G}_1 \leftarrow \mathbb{F}^{K+k-|S'|\times|S'|}$, $\mathbf{G}_2 \leftarrow \mathbb{F}^{|S'|\times|S'|}$, $\mathbf{R} \leftarrow \mathbb{F}^{|S'|\times K+k-|S'|}$

Compute a matrix $\mathbf{T} \in \mathbb{F}^{|S'|\times|S'|}$ such that $(\mathbf{G}_1^\top \mathbf{R}^\top - \mathbf{G}_2^\top)\mathbf{T} = \mathbf{I}_{|S'|}$. Abort if such a matrix does not exist

Denote by $\overline{\mathbf{G}}_{\overline{S'}}$ the matrix containing the first $K + k - |S'|$ rows of $[\mathbf{G}_{\overline{S'}}]$

Output $\mathsf{ck}_{\mathsf{obl}} = [\mathbf{G}^*] = \begin{pmatrix} [\mathbf{G}_1] & [\overline{\mathbf{G}}_{\overline{S'}}] \\ [\mathbf{G}_2] & \mathbf{R}[\overline{\mathbf{G}}_{\overline{S'}}] \end{pmatrix} \mathbf{P}_{S'}$ and $\tau_{\mathsf{obl}} = \mathbf{T}^* = \begin{pmatrix} \mathbf{R}^\top\mathbf{T} \\ -\mathbf{T} \end{pmatrix}$

Com(ck, $\mathbf{x}$)**:** Parse ck $= [\mathbf{G}]$ and output $[\mathbf{c}] = [\mathbf{G}]\mathbf{x}$

Extract($\tau, [\mathbf{c}]_\mu$)**:** Output $[\mathbf{x}_S] = \mathbf{T}_S^\top[\mathbf{c}]$

Proof. First, we show that CS′ is an SSB commitment. For index-hiding we can use a standard hybrid argument to show that the concatenation of K commitment keys are indeed indistinguishable. Somewhere Statistical Binding and G-extractability of CS′ follow from the respective properties of CS. Indeed, for the former, note that each individual commitment is statistically binding in one coordinate, and for a commitment-opening to verify, all commitments are checked w.r.t. to the same opening; thus, effectively the commitment is statistically binding in the set S. For the latter, we use the same argument and the fact that the extractor of CS can G-extract each value independently.

For oblivious trapdoor generation, note that the srs output by OblKeyGen follow exactly the same distribution as the one output by KeyGen as well as a valid trapdoor for S'. □

Next, we present MPed a more efficient SSB commitment scheme family with oblivious trapdoor generation. The construction is presented in Fig. 4.3. The scheme is parameterized by a group description gk, the message space is $\mathbb{F}^n$ and we extract $[\mathbf{x}_S]$. The construction is essentially the one given in [FLPS20], which in turn is a generalization of the so called *Multi-Pedersen commitments* from [GHR15], with a minor change in the key generation algorithm.

For simplicity, we describe the oblivious key generation algorithm in terms of the permutation

$\mathbf{P}_S$ while it is not really needed. Indeed, it only needs to randomly sample itself the columns corresponding to S' and sample the lower rows as a random combination of upper rows or columns in $\overline{S'}$.

In [FLPS20] it is shown that the Index Set Hiding property can be reduced to DDH with a security lost of $2\log K$ when $\mathbf{G}_0$ is uniform using the results of [Vil12]. In our case $\mathbf{G}_0$ it is not completely uniform as some part depends on $\mathcal{D}_k$. While it seems still possible to use [Vil12], we use for simplicity a naive hybrid argument at the cost of a less tight reduction. Although the security lost is $2K$ instead of $2\log K$, in general K is small (constant in our instantiations) and hence it doesn't make much difference.

Theorem 12. *Construction* MPed *of Fig. 4.3 is an SSB commitment scheme. It is somewhere statistically binding and $\mathbb{G}$-extractable with knowledge error $\frac{K}{p}$ and Index Set Hiding with distinguishing advantage at most $2K \cdot \mathcal{A}_{\mathsf{MDDH}\text{-}\mathcal{D}_k}(\mathcal{D})$, where $\mathcal{D}$ is a PPT adversary against the $\mathcal{D}_k$-*MDDH *assumption.*

Proof. We first show that KeyGen aborts only with probability $\frac{K}{p}$. Let $\mathbf{G}_0^\perp$ be a matrix whose columns are a basis of the kernel of $\mathbf{G}_0^\top$. Since $\mathbf{G}_0$ is uniformly distributed, by the Schwartz-Zippel lemma, $\mathbf{G}_0$ has rank $K + k - |S|$ with probability at least $1 - \frac{K+k-|S|}{p}$. Now, consider the matrix $\mathbf{G}_S^\top \mathbf{G}_0^\perp$. Again, by the Schwartz-Zippel lemma and the fact that $\mathbf{G}_S$ is uniformly distributed, this matrix has rank $|S|$ with probability at least $1 - \frac{|S|}{p}$, and thus, it is invertible. Let $\mathbf{T}$ be its inverse. This matrix exists except with probability $\frac{K+k-|S|+|S|}{p} = \frac{K+k}{p}$. Now, set $\mathbf{T}_S = \mathbf{G}_0^\perp \mathbf{T}$. We have that $\mathbf{G}_S^\top \mathbf{T}_S = \mathbf{G}_S^\top \mathbf{G}_0^\perp \mathbf{T}' = \mathbf{I}_{|S|}$ and $\mathbf{G}_0^\top \mathbf{T}_S = \mathbf{G}_0^\top \mathbf{G}_0^\perp \mathbf{T}' = \mathbf{0}_{K+k-|S|\times S}$, which concludes the proof.

Index Set Hiding. Consider the following sequence of hybrid games.

- $\mathsf{Game}_0^{\mathcal{D}}$: In this game we sample $(\mathsf{ck}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa, \mathsf{gk}, n, K, S_0)$ and output $\mathcal{D}(\mathsf{ck})$.
- $\mathsf{Game}_1^{\mathcal{D}}$: In this game we sample $(\mathsf{ck}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa, \mathsf{gk}, n, K, \emptyset)$ and output $\mathcal{D}(\mathsf{ck})$.
- $\mathsf{Game}_2^{\mathcal{D}}$: In this game we sample $(\mathsf{ck}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa, \mathsf{gk}, n, K, S_1)$ and output $\mathcal{D}(\mathsf{ck})$.

Noting that in Game_0 and in Game_1, the difference in the distributions of ck is that in the former $\mathbf{G}_{S_0}$ is uniform, while in the later $\mathbf{G}_{S_0} = \mathbf{G}_0 \mathbf{\Gamma}_{S_0}$, where $\mathbf{\Gamma}_{S_0} \in \mathbb{F}^{K+k-|S|\times|S_0|}$. Using a standard hybrid argument, we can bound the advantage of distinguishing these games by $|S_0| \leq K$ times the advantage of breaking the $\mathcal{G}_0$-MDDH assumption. It is not hard to see that the $\mathcal{G}_0$-MDDH can be reduced (without security lost) to the $\mathcal{D}_k$-MDDH assumption. We conclude that the advantage of distinguishing Game_0 and Game_1 can be bounded by $K \cdot \mathcal{A}_{\mathcal{D}_k\text{-}\mathsf{MDDH}}$. The same argument applies to Game_1 and in Game_2.

Somewhere Statistically Binding. Finally we show the somewhere statistically binding and extractability property. Let $(\mathsf{ck}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{gk}, n, K, S)$ be the sampled key and

$\mathbf{G}_S, \mathbf{G}_0, \mathbf{\Gamma}$ the values defining it. Conditioned on KeyGen not failing, which only happens with probability at most $1 - \frac{K}{p}$, the matrix $\mathbf{T}_S \in \mathbb{F}^{K+k\times|S|}$ satisfies $\mathbf{T}_S^\top \mathbf{G} = \mathbf{\Sigma}_S \mathbf{P}_S$.

Now let $\mathbf{x}, \mathbf{x}' \in \mathbb{F}^n$. For extractability, note that

$$\mathbf{T}^\top \mathsf{Com}([\mathbf{G}], \mathbf{x}) = \mathbf{T}^\top [\mathbf{G}]\mathbf{x} = [\mathbf{\Sigma}_S \mathbf{P}_S]\mathbf{x} = [\mathbf{x}_S]$$

Additionally, if $\mathsf{c}([\mathbf{G}], \mathbf{x}) = \mathsf{c}([\mathbf{G}], \mathbf{x}')$ and we multiply by $\mathbf{T}^\top$ on both sides, we get that $\mathbf{x}_S = \mathbf{x}'_S$. □

In the next Theorem we assume $\mathcal{D}_k$ outputs full rank matrices with overwhelming probability. Note that this is true for most matrix distributions such as the uniform and the linear family.

Theorem 13. *Construction* MPed *of Fig. 4.3 satisfies Oblivious Trapdoor Generation. Furthermore, for all even unbounded $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2)$, against oblivious trapdoor generation, $\mathcal{A}^{\mathsf{MPed}}_{\mathsf{obl}}(\mathcal{D}) \leq \frac{K}{p}$.*

Proof. Let $K \leq n$ and $S' \subseteq S \subseteq [n]$. We first show that the oblivious key follows exactly the same distribution as the original key. Let $\mathsf{ck} := [\mathbf{G}]$ be the output of $\mathsf{KeyGen}(\mathsf{gk}, n, K, S)$ and $\mathsf{ck}_{\mathsf{obl}} = [\mathbf{G}^*]_\mu$ be the output of $\mathsf{OblKeyGen}(\mathsf{gk}, n, K, S', [\mathbf{G}])$. We can write ck as $\mathbf{G} = \left((\mathbf{G}_{S'} \mid \mathbf{G}_{S'|S})\mathbf{P}_{S'|S} \quad \mathbf{G}_0\mathbf{\Gamma}\right)\mathbf{P}_S$.

Let $\overline{\mathbf{G}}_{S'|S} \in \mathbb{F}_p^{K+k-|S'|\times K-|S'|}$, $\underline{\mathbf{G}}_{S'|S} \in \mathbb{F}^{|S'|\times K-|S'|}$, $\overline{\mathbf{G}}_0 \in \mathbb{F}^{K+k-|S'|\times k}$, $\underline{\mathbf{G}}_0 \in \mathbb{F}_p^{|S'|\times k}$, such that $\mathbf{G}_{S'|S} = \begin{pmatrix}\overline{\mathbf{G}}_{S'|S}\\ \underline{\mathbf{G}}_{S'|S}\end{pmatrix}, \mathbf{G}_0 = \begin{pmatrix}\overline{\mathbf{G}}_0\\ \underline{\mathbf{G}}_0\end{pmatrix}$. We claim that there exists a matrix $\mathbf{R} \in \mathbb{F}^{|S'|\times K+k-|S'|}$, uniformly distributed, such that $\left(\underline{\mathbf{G}}_{S'|S}|\underline{\mathbf{G}}_0\right) = \mathbf{R}\left(\overline{\mathbf{G}}_{S'|S}|\overline{\mathbf{G}}_0\right)$ as in the output of OblKeyGen. If this is the case, the distributions of ck output by KeyGen and $\mathsf{ck}_{\mathsf{obl}}$ output by OblKeyGen are identical, since we can write

$$\begin{aligned}
\mathbf{G} &= \left(\mathbf{G}_{S'} \quad \left(\begin{pmatrix}\overline{\mathbf{G}}_{S'|S} & \overline{\mathbf{G}}_0\\ \underline{\mathbf{G}}_{S'|S} & \underline{\mathbf{G}}_0\end{pmatrix} \quad \begin{pmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{\Gamma}\end{pmatrix}\right)\mathbf{P}_{S'|S}\right)\mathbf{P}_S\\
&= \left(\mathbf{G}_{S'} \quad \left(\begin{pmatrix}\overline{\mathbf{G}}_{S'|S} & \overline{\mathbf{G}}_0\\ \mathbf{R}\left(\overline{\mathbf{G}}_{S'|S} & \overline{\mathbf{G}}_0\right)\end{pmatrix} \quad \begin{pmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{\Gamma}\end{pmatrix}\right)\mathbf{P}_{S'|S}\right)\mathbf{P}_S\\
&= \begin{pmatrix}\mathbf{G}_{S'} & \overline{\mathbf{G}}_{\overline{S}}\\ & \mathbf{R}\overline{\mathbf{G}}_{\overline{S}}\end{pmatrix}\mathbf{P}_S
\end{aligned}$$

First we show that the matrix $(\overline{\mathbf{G}}_{S'|S}|\overline{\mathbf{G}}_0)$ is full rank with overwhelming probability. Indeed, $\overline{\mathbf{G}}_0 = \begin{pmatrix}\mathbf{A}\\ \overline{\mathbf{W}}\mathbf{A}\end{pmatrix}$, where $\mathbf{A} \leftarrow \mathcal{D}_k, \overline{\mathbf{W}} \leftarrow \mathbb{F}^{K-1-|S'|\times k+1}$, and it has rank k. By the fact that $\overline{\mathbf{G}}_{S'|S}$ is uniform, using the Schwartz-Zippel lemma we get that $(\overline{\mathbf{G}}_{S'|S}|\overline{\mathbf{G}}_0)$ has rank

Figure 4.4 KMPed SSB commitment schemes parametrized by the matrix distributions $\mathcal{D}_k$. We denote gk_μ the description of $\mathbb{G}_\mu$.

$\mathsf{KeyGen}(\mathsf{gk}, n, K, S)$:

Let $\mathbf{G}, \mathbf{T_G} \leftarrow \mathsf{MPed.KeyGen}(\mathsf{gk}_1, n, K, S)$

Let $\mathbf{H}, \mathbf{T_H} \leftarrow \mathsf{MPed.KeyGen}(\mathsf{gk}_2, n, K, S)$

$\mathbf{Z} \leftarrow \mathbb{F}(K+k)^2 \times n^2$

Define the three keys

$$\begin{array}{rrr} \mathsf{ck}_1 = [\mathbf{G}]_1, & \tau_1 = \mathbf{T_G}, & \mathsf{aux}_1 = ([\mathbf{H}]_2, [\mathbf{G}\otimes\mathbf{H}-\mathbf{Z}]_1, [\mathbf{Z}]_2) \\ \mathsf{ck}_2 = [\mathbf{H}]_2, & \tau_2 = \mathbf{T_H}, & \mathsf{aux}_2 = ([\mathbf{G}]_1, [\mathbf{G}\otimes\mathbf{H}-\mathbf{Z}]_1, [\mathbf{Z}]_2) \\ \mathsf{ck}_s = ([\mathbf{G}\otimes\mathbf{H}-\mathbf{Z}]_1, [\mathbf{Z}]_2) & \tau_s = \mathbf{T_H}\otimes\mathbf{T_H}, & \mathsf{aux}_s = ([\mathbf{G}]_1, [\mathbf{H}]_2) \end{array}$$

$K+k-|S'|$ except with probability $\frac{K-|S'|}{p}$. This means that the matrix is invertible and we can set $\mathbf{R} = (\underline{\mathbf{G}}_{S'|S}|\underline{\mathbf{G}}_0)(\overline{\mathbf{G}}_{S'|S}|\overline{\mathbf{G}}_0)^{-1}$. Furthermore, both $\underline{\mathbf{G}}_{S'|S}$ and $\underline{\mathbf{G}}_0 = \underline{\mathbf{W}}\mathbf{A}$ are uniform, the latter since $\underline{\mathbf{W}} \in \mathbb{F}^{|S'|\times k+1}$ is uniformly distributed and $\mathbf{A}$ is full rank, and the former by construction.

To conclude the proof, we to show that the trapdoor output by $\mathsf{OblKeyGen}(\mathsf{gk}, n, K, S', [\mathbf{G}])$ is correct w.r.t $\mathsf{ck}_{\mathsf{obl}}$, that is $\mathbf{T}^{*\top}\mathbf{G}^* = \mathbf{\Sigma}_{S'}$. By a simple calculation,

$$\mathbf{T}^{*\top}\mathbf{G}^* = \begin{pmatrix}\mathbf{T}^\top\mathbf{R} & -\mathbf{T}\end{pmatrix}\begin{pmatrix}\mathbf{G}_1 & \overline{\mathbf{G}}_{\overline{S'}} \\ \mathbf{G}_2 & \mathbf{R}\overline{\mathbf{G}}_{\overline{S'}}\end{pmatrix} = \begin{pmatrix}\mathbf{T}^\top(\mathbf{R}\mathbf{G}_1 - \mathbf{G}_2) & \mathbf{T}^\top\mathbf{R}\overline{\mathbf{G}}_{\overline{S'}} - \mathbf{T}^\top\mathbf{R}\overline{\mathbf{G}}_{\overline{S'}}\end{pmatrix} = \begin{pmatrix}\mathbf{I}_{|S'|} & \mathbf{0}\end{pmatrix} = \mathbf{\Sigma}_{S'}$$

where $\mathbf{T}^\top(\mathbf{R}\mathbf{G}_1 - \mathbf{G}_2) = \mathbf{I}_{S'}$ by construction. □

In the next sections we assume that KeyGen and OblKeyGen do not abort. This is w.l.o.g. since we can always re-sample values when an abort happens. Note that in this case, the keys of both KeyGen and OblKeyGen are "somewhere perfectly binding".

Finally, we present in Fig 4.4 a construction that is based on Kronecker product distributions. Consider the values $[\mathbf{G}]_1, [\mathbf{H}]_2, [\mathbf{G}\otimes\mathbf{H}-\mathbf{Z}]_1, [\mathbf{Z}]_2$, where $[\mathbf{G}]_1, [\mathbf{H}]_2$ are computed according to MPed and $\mathbf{Z}$ is uniformly distributed. From this distribution, we can derive three different commitment keys: $[\mathbf{G}]_1$ for committing to $\mathbb{G}_1$, $[\mathbf{H}]_2$ for committing to $\mathbb{G}_2$ and a split key $[\mathbf{G}\otimes\mathbf{H}-\mathbf{Z}]_1$, $[\mathbf{Z}]_2$. We refer to the first two keys as the *source* keys and to the latter as the *target* key. We only describe the key generation algorithm since Com and Extract are derived by the algebraic structure of the scheme. For simplicity, we only consider the case we will be using later: "combining" commitments with the same parameters n, K and extractable at the same set S. The construction, however, generalizes to arbitrary parameters and sets.

The source commitments are identical to MPed construction for $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. The only difference is that the adversary also knows the "split" part of the key $[\mathbf{G}\otimes\mathbf{H}-\mathbf{Z}]_1$, $[\mathbf{Z}]_2$ and we need to show that the properties are preserved. We next show that all three

derived constructions are SSB commitment schemes even with the auxiliary information the adversary is given for each. The source schemes are extractable at S and the target scheme at $S^2 = \{(s_1 - 1)n + s_2\}_{s_1,s_2 \in S}$.

Theorem 14. *If we instantiate* KMPed *using* MPed *under any distribution $\mathcal{D}_k$ that satisfies $\mathcal{D}_k$-*MDDH *assumption in both groups then the source constructions are algebraic SSB commitment scheme and the target construction is a split algebraic commitment key. Furthermore, all derived schemes have oblivious trapdoor generator.*

Proof.

Somewhere Statistically Binding and G-Extractability. The source constructions are locally extractable by the local extractability of MPed. We show that the target construction is also extractable at S^2. Let $\mathbf{Q}_1 = \mathbf{G} \otimes \mathbf{H} - \mathbf{Z}$ and $\mathbf{Q}_2 = \mathbf{Z}$ and assume that $\mathbf{z}, \mathbf{z}' \in \mathbb{F}^{n^2}$ satisfy $\mathsf{KMPed}_t.\mathsf{Com}(\mathsf{ck}, \mathbf{z}) = \mathsf{KMPed}_t\mathsf{Com}(\mathsf{ck}, \mathbf{z}')$. This means that $(\mathbf{G} \otimes \mathbf{H})\mathbf{z} = (\mathbf{G} \otimes \mathbf{H})\mathbf{z}'$. Therefore,

$$\begin{aligned}
\mathbf{0} &= (\mathbf{T_G} \otimes \mathbf{T_H})(\mathbf{G} \otimes \mathbf{H})(\mathbf{z} - \mathbf{z}') \\
&= (\mathbf{T_G}\mathbf{G}) \otimes (\mathbf{T_H}\mathbf{H})(\mathbf{z} - \mathbf{z}') \\
&= (\mathbf{\Sigma}_S\mathbf{P}_S) \otimes (\mathbf{\Sigma}_S\mathbf{P}_S)(\mathbf{z} - \mathbf{z}') \\
&= (\mathbf{\Sigma}_S \otimes \mathbf{\Sigma}_S)(\mathbf{P}_S \otimes \mathbf{P}_S)(\mathbf{z} - \mathbf{z}') \\
&= \mathbf{z}_{S,S} - \mathbf{z}'_{S,S} \\
&= \mathbf{z}_{S^2} - \mathbf{z}'_{S^2}
\end{aligned}$$

Note that this also shows that the trapdoors correctly extracts a correct splitting of $\mathbf{z}_S$ from $\mathsf{KMPed}_t.\mathsf{Com}(\mathsf{ck}, \mathbf{z})$.

Index Set Hiding. For index set hiding, it is enough to show that the collective key

$$[\mathbf{G}]_1, [\mathbf{H}]_2, [\mathbf{G} \otimes \mathbf{H} + \mathbf{Z}]_1, [\mathbf{Z}]_2$$

is indistinguishable for different values $(S_1, S_2) \neq (S'_1, S'_2)$. The result follows from the indistinguishability of the following distributions (this is essentially part of the proof in [GHR15, Theorem 6]).

1. $[\mathbf{G}]_1, [\mathbf{H}]_2, [\mathbf{G} \otimes \mathbf{H} + \mathbf{Z}]_1, [-\mathbf{Z}]_2$, — $\mathbf{G}, \mathbf{H}$ binding at S_1, S_2
2. $[\mathbf{G}]_1, [\mathbf{H}]_2, [\mathbf{G}]_1 \otimes \mathbf{H} + [\mathbf{Z}]_1, [-\mathbf{Z}]_2$ — $\mathbf{G}, \mathbf{H}$ binding at S_1, S_2
3. $[\mathbf{G}]_1, [\mathbf{H}]_2, [\mathbf{G}]_1 \otimes \mathbf{H} + [\mathbf{Z}]_1, [-\mathbf{Z}]_2$ — $\mathbf{G}, \mathbf{H}$ binding at S'_1, S_2
4. $[\mathbf{G}]_1, [\mathbf{H}]_2, [\mathbf{Z}]_1, \mathbf{G} \otimes [\mathbf{H}]_2 - [\mathbf{Z}]_2$ — $\mathbf{G}, \mathbf{H}$ binding at S'_1, S_2
5. $[\mathbf{G}]_1, [\mathbf{H}]_2, [\mathbf{Z}]_1, \mathbf{G} \otimes [\mathbf{H}]_2 - [\mathbf{Z}]_2$ — $\mathbf{G}, \mathbf{H}$ binding at S'_1, S'_2
6. $[\mathbf{G}]_1, [\mathbf{H}]_2, [\mathbf{G} \otimes \mathbf{H} + \mathbf{Z}]_1, [-\mathbf{Z}]_2$ — $\mathbf{G}, \mathbf{H}$ binding at S'_1, S'_2

Perfect indistinguishability between distributions 1-2, 3-4 and 5-6 follows from the fact that always both distributions are uniformly distributed conditioned on their sum being equal to $\mathbf{G} \otimes \mathbf{H}$. Computational indistinguishability of distributions 2-3 and 4-5 follows from the ISH of MPed_1 and MPed_2 respectively.

Oblivious Trapdoor Generation. We will show how to simultaneously sample oblivious keys for all constructions. We construct an oblivious key generation algorithm as follows.

$\mathsf{KMPed.OblKeyGen}(\mathsf{gk}, n, K, S, ([\mathbf{Q}_1]_1, [\mathbf{Q}_2]_2, [\mathbf{G}]_1, [\mathbf{H}]_2))$**:**

1. Get oblivious keys and trapdoors for $\mathbf{G}, \mathbf{H}$ using $\mathsf{MPed}_\mu.\mathsf{OblKeyGen}$.

$$([\mathbf{G}^*]_1, \mathbf{T}_1) \leftarrow \mathsf{MPed}_1.\mathsf{OblKeyGen}(\mathsf{gk}, n, K, S, [\mathbf{G}]_1)$$

$$([\mathbf{H}^*]_2, \mathbf{T}_2) \leftarrow \mathsf{MPed}_2.\mathsf{OblKeyGen}(\mathsf{gk}, n, K, S, [\mathbf{H}]_2)$$

   and use the random coins of $\mathsf{OblKeyGen}$ to retrieve $\mathbf{G}^*_{S_1}, \mathbf{R}_1$ and $\mathbf{H}^*_{S_1}, \mathbf{R}_2$ such that

$$[\mathbf{G}^*]_1 = \begin{pmatrix} [\mathbf{G}^*_S]_1 & [\overline{\mathbf{G}}_{\overline{S}}]_1 \\ \mathbf{R}_1[\overline{\mathbf{G}}_{\overline{S}}]_1 \end{pmatrix} \mathbf{P}_S \text{ and } \mathbf{H}^* = \begin{pmatrix} [\mathbf{H}^*_S]_2 & [\overline{\mathbf{H}}_{\overline{S}}]_2 \\ \mathbf{R}[\overline{\mathbf{H}}_{\overline{S}}]_2 \end{pmatrix} \mathbf{P}_S,$$

   as defined in Fig. 4.3.

2. Let $[\mathbf{A}_1]_1, [\mathbf{A}_2]_2$ be the matrices containing the first $(K + k - |S|)(K + k)$ rows of the matrices $[(\mathbf{Q}_1)_{\overline{S^2}}]_1$ and $[(\mathbf{Q}_2)_{\overline{S^2}}]_2$.
3. Let $\mathbf{\Pi}_1$ and $\mathbf{\Pi}_2$ the permutation matrices of Fact 3 for matrices with $(K+k-|S|)^2$ rows, and $(n - |S|)^2$ columns.
4. Define $[\mathbf{B}_1]_1$ and $[\mathbf{B}_2]_2$ as the matrices of the first $(K+k-|S|)^2$ columns of $\mathbf{\Pi}_1^\top[\mathbf{A}_1]_1\mathbf{\Pi}_2^\top$ and $\mathbf{\Pi}_1^\top[\mathbf{A}_2]_2\mathbf{\Pi}_2^\top$, respectively.
5. Redefine $\mathbf{A}_1, \mathbf{A}_2$ as

$$[\mathbf{A}^*_1]_1 = \mathbf{\Pi}_1 \begin{pmatrix} [\mathbf{B}_1]_1 \\ (\mathbf{R}_2 \otimes \mathbf{I}_{K+k-|S|})[\mathbf{B}_1]_1 \end{pmatrix} \mathbf{\Pi}_2 \text{ and } [\mathbf{A}^*_2]_2 = \mathbf{\Pi}_1 \begin{pmatrix} [\mathbf{B}_2]_2 \\ (\mathbf{R}_2 \otimes \mathbf{I}_{K+k-|S|})[\mathbf{B}_2]_2 \end{pmatrix} \mathbf{\Pi}_2$$

6. Pick uniform matrices

$$\mathbf{Z}_1 \leftarrow \mathbb{F}^{(K+k)\times n|S|}, \qquad \mathbf{Z}_2 \leftarrow \mathbb{F}^{(K+k)\times(n-|S|)|S|}, \qquad \mathbf{Z}_3 \leftarrow \mathbb{F}^{(K+k)\times(n-|S|)(n-|S|}$$

7. Define split key

$$[\mathbf{Q}^*_1]_1 = \begin{pmatrix} \mathbf{Z}_1 & [\mathbf{G}^*_{\overline{S}}]_1 \otimes \mathbf{H}^*_S + [\mathbf{Z}_2]_1 & \begin{pmatrix} [\mathbf{A}^*_1]_1 \\ (\mathbf{R}_1 \otimes \mathbf{I}_{K+k})[\mathbf{A}^*_1]_1 \end{pmatrix} + [\mathbf{Z}_3]_1 \end{pmatrix} (\mathbf{P}_S \otimes \mathbf{P}_S)$$

$$[\mathbf{Q}^*_2]_2 = \begin{pmatrix} \mathbf{G}^*_S \otimes [\mathbf{H}^*]_2 - [\mathbf{Z}_1]_2 & -[\mathbf{Z}_2]_2 & \begin{pmatrix} [\mathbf{A}^*_2]_2 \\ (\mathbf{R}_1 \otimes \mathbf{I}_{K+k})[\mathbf{A}^*_2]_2 \end{pmatrix} - [\mathbf{Z}_3]_2 \end{pmatrix} (\mathbf{P}_S \otimes \mathbf{P}_S)$$

8. Return $([\mathbf{G}^*]_1, [\mathbf{H}^*]_2, [\mathbf{Q}^*_1]_1, [\mathbf{Q}^*_2]_2, \tau = \mathbf{T}_1 \otimes \mathbf{T}_2)$.

Next we show that the key is correctly distributed. Since MPed_1 and MPed_2 are both oblivious SSB commitments, it holds that $[\mathbf{G}^*]_1, [\mathbf{H}^*]_2$ have the correct distribution. It

remains to show that $\mathbf{Q}^* = \mathbf{Q}_1^* + \mathbf{Q}_2^* = \mathbf{G}^* \otimes \mathbf{H}^*$. This is enough since if this holds, the commitment key $[\mathbf{Q}_1^*]_1, [\mathbf{Q}_2^*]_2$ consists of two uniform matrices, conditioned on their sum equaling $\mathbf{G}^* \otimes \mathbf{H}^*$, and this is the distribution of the honest key as well.

First, observe that omitting the permutations, the shape of the matrix $\mathbf{G}^* \otimes \mathbf{H}^*$ will be

$$\mathbf{G}^* \otimes \mathbf{H}^* = \begin{pmatrix} \mathbf{G}_S^* & \mathbf{G}_{\overline{S}}^* \end{pmatrix} \otimes \begin{pmatrix} \mathbf{H}_S^* & \mathbf{H}_{\overline{S}}^* \end{pmatrix} = \begin{pmatrix} \mathbf{G}_S^* \otimes \mathbf{H}^* & \mathbf{G}_{\overline{S}}^* \otimes \mathbf{H}_S^* & \mathbf{G}_{\overline{S}}^* \otimes \mathbf{H}_{\overline{S}}^* \end{pmatrix}$$

By construction, the first two parts of the matrix $\mathbf{Q}^*$ we construct are consistent with $\mathbf{G}_S^* \otimes \mathbf{H}^*$ and $\mathbf{G}_{\overline{S}}^* \otimes \mathbf{H}_S^*$ respectively.

It remains to show that the same hold for the third part of $\mathbf{Q}_1^*, \mathbf{Q}_2^*$. Noting that these columns correspond to the indices not in S^2, we denote them $\mathbf{Q}^*_{1,\overline{S^2}}, \mathbf{Q}^*_{2,\overline{S^2}}$. First, note that for the initial matrices $\mathbf{Q}_1, \mathbf{Q}_2$ it holds that $\mathbf{Q}_{1,\overline{S^2}} + \mathbf{Q}_{2,\overline{S^2}} = \begin{pmatrix} \overline{\mathbf{G}}_{\overline{S}} \otimes \mathbf{H}_{\overline{S}} \\ \underline{\mathbf{G}}_{\overline{S}} \otimes \mathbf{H}_{\overline{S}} \end{pmatrix}$ Furthermore, $\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2 = \overline{\mathbf{G}}_{\overline{S}} \otimes \mathbf{H}_{\overline{S}}$. It follows that

$$\mathbf{\Pi}_1^\top \mathbf{A} \mathbf{\Pi}_2^\top = \mathbf{\Pi}_1^\top \mathbf{\Pi}_1 \mathbf{H}_{\overline{S}_2} \otimes \overline{\mathbf{G}}_{\overline{S}_1} \mathbf{\Pi}_2 \mathbf{\Pi}_2^\top = \begin{pmatrix} \overline{\mathbf{H}}_{\overline{S}_2} \otimes \overline{\mathbf{G}}_{\overline{S}_1} \\ \underline{\mathbf{H}}_{\overline{S}_2} \otimes \overline{\mathbf{G}}_{\overline{S}_1} \end{pmatrix}$$

and hence $\mathbf{B} = \mathbf{B}_1 + \mathbf{B}_2 = \overline{\mathbf{H}}_{\overline{S}} \otimes \overline{\mathbf{G}}_{\overline{S}}$. Finally we have that

$$\begin{aligned}
\mathbf{Q}^*_{\overline{S},\overline{S}} &= \begin{pmatrix} \mathbf{A}_1^* + \mathbf{A}_2^* \\ (\mathbf{R}_1 \otimes \mathbf{I}_{K+k})(\mathbf{A}_1^* + \mathbf{A}_2^*) \end{pmatrix} = \begin{pmatrix} \mathbf{\Pi}_1 \begin{pmatrix} \mathbf{B}_1 + \mathbf{B}_2 \\ (\mathbf{R}_2 \otimes \mathbf{I}_{K+k-|S|})(\mathbf{B}_1 + \mathbf{B}_2) \end{pmatrix} \mathbf{\Pi}_2 \\ (\mathbf{R}_1 \otimes \mathbf{I}_{K+k})(\mathbf{A}_1^* + \mathbf{A}_2^*) \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{\Pi}_1 \begin{pmatrix} \overline{\mathbf{H}}_{\overline{S}} \otimes \overline{\mathbf{G}}_{\overline{S}} \\ (\mathbf{R}_2 \otimes \mathbf{I}_{K+k-|S_1|})(\overline{\mathbf{H}}_{\overline{S}} \otimes \overline{\mathbf{G}}_{\overline{S}}) \end{pmatrix} \mathbf{\Pi}_2 \\ (\mathbf{R}_1 \otimes \mathbf{I}_{K+k})(\mathbf{A}_1^* + \mathbf{A}_2^*) \end{pmatrix} = \begin{pmatrix} \mathbf{\Pi}_1 \begin{pmatrix} \overline{\mathbf{H}}_{\overline{S}} \\ \mathbf{R}_2 \overline{\mathbf{H}}_{\overline{S}} \end{pmatrix} \otimes \overline{\mathbf{G}}_{\overline{S}} \mathbf{\Pi}_2 \\ (\mathbf{R}_1 \otimes \mathbf{I}_{K+k})(\mathbf{A}_1^* + \mathbf{A}_2^*) \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{\Pi}_1 \mathbf{H}^*_{\overline{S}} \otimes \overline{\mathbf{G}}_{\overline{S}} \mathbf{\Pi}_2 \\ (\mathbf{R}_1 \otimes \mathbf{I}_{K+k})(\mathbf{A}_1^* + \mathbf{A}_2^*) \end{pmatrix} = \begin{pmatrix} \overline{\mathbf{G}}_{\overline{S}_1} \otimes \mathbf{H}^*_{\overline{S}_2} \\ (\mathbf{R}_1 \otimes \mathbf{I}_{K_2+k})(\overline{\mathbf{G}}_{\overline{S}_1} \otimes \mathbf{H}^*_{\overline{S}_2}) \end{pmatrix} \\
&= \mathbf{G}^*_{\overline{S}} \otimes \mathbf{H}^*_{\overline{S}}.
\end{aligned}$$

□

4.4 Quasi-Arguments with Pre-processing

In this section we introduce an extension of Quasi Arguments as defined in [KPY19] which adds support for language dependent srs or pre-processing such as the so called QA-NIZK arguments [JR13]. Additionally we use different languages for completeness and local soundness, i.e. promise problems, to incorporate the "knowledge transfer" soundness of [GR19].

Following [JR13], languages are parametrized by $\rho \in \mathcal{L}_{\mathsf{par}}$ and ρ sampled from some distribution $\mathcal{D}_{\mathsf{par}}$. We say sf that $\mathcal{D}_{\mathsf{par}}$ is witness samplable if ρ can be efficiently sampled together with a witness θ for $\rho \in \mathcal{L}_{\mathsf{par}}$. We simply write $(\theta, \rho) \leftarrow \mathcal{D}_{\mathsf{par}}$. Each $\rho \in \mathcal{L}_{\mathsf{par}}$ defines a language $\mathcal{L}_\rho$ with the corresponding relations $\mathcal{R}_\rho^{\mathsf{yes}}$, that is $\mathcal{L}_\rho = \{\mathsf{x} \mid \exists \mathsf{w} \text{ s.t. } (\mathsf{x}, \mathsf{w}) \in \mathcal{R}_\rho^{\mathsf{yes}}\}$. After the language is fixed there is a (language dependent) prepossessing stage where a common reference string is generated. Going a step forward, we would like our statements to be commitments and that $\mathcal{R}_\rho^{\mathsf{yes}}$ puts some restriction on the commitment opening. Since we will be using SSB commitments, the language parameter must contain the SSB commitment key. Therefore, we assume distribution $\mathcal{D}_{\mathsf{par}}$ receives as input $d \in \mathbb{N}$ (the size of the opening), a locality parameter $K \leq d$ and a set $S \subseteq [d]$. It will be useful to define $\mathcal{L}_\rho^{\mathsf{yes}} = \mathcal{L}_\rho$ and $\mathcal{L}_\rho^{\mathsf{no}}$ the complement of $\mathcal{L}_\rho^{\mathsf{yes}}$, and similarly define $\mathcal{R}_\rho^{\mathsf{yes}}$ and $\mathcal{R}_\rho^{\mathsf{no}}$. Traditional arguments of knowledge require that from any accepting statement and proof pair one can extract a witness w such that $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}_\rho^{\mathsf{no}}$ only with negligible probability. In a quasi-argument of knowledge only a small part of the witness w_S is extracted and $(\mathsf{x}, \mathsf{w}_S) \in \mathcal{R}_{\rho,S}^{\mathsf{yes}}$ with overwhelming probability, where $\mathcal{R}_{\rho,S}^{\mathsf{yes}}$ is a "local version" of $\mathcal{R}_\rho^{\mathsf{yes}}$.[12]

Our final addition is support for arguments of knowledge transfer (AoKT) [GR19]. In a nutshell, an AoKT enables to "succinctly reuse" an AoK of the opening of some commitment C for constructing another AoK for commitment D. That is, given an opening w for C, it enables to give a succinct proof that D opens to $g(\mathsf{w})$. Importantly, AoKTs can be based on falsifiable assumptions. Following [GR19], $\rho \in \mathcal{L}_{\mathsf{par}}$ defines languages $\mathcal{L}_\rho^{\mathsf{yes}}$ and $\mathcal{L}_\rho^{\mathsf{no}}$, with $\mathcal{L}_\rho^{\mathsf{no}}$ not necessarily the complement of $\mathcal{L}_\rho^{\mathsf{yes}}$ (i.e. a promise problem), with their corresponding relations $\mathcal{R}_\rho^{\mathsf{yes}}$ and $\mathcal{R}_\rho^{\mathsf{no}}$. For no instances, the adversary provides a promise w^* for x. In [GR19] $\mathsf{x} = (C, D)$ and $(C, D, \mathsf{w}^*) \in \mathcal{L}_\rho^{\mathsf{no}}$ if w^* is an opening for C but $g(\mathsf{w}^*)$ is not an opening for D. In our instantiations x will be two SSB commitments to $C_1, \ldots, C_d$ and $D_1, \ldots, D_d$ such that C_i opens to w and D_i to $g_i(\mathsf{w})$. From the two SSB commitments we can extract C_S and D_S. Furthermore, C_i and D_i might not be extractable (actually, they will be Pedersen commitments) an hence the extractor can only compute $f(\mathsf{w}, S) = \{\mathsf{Com}(\mathsf{ck}_i, \mathsf{w}) : i \in S\}$.

We define the yes and no languages as

$$\mathcal{L}_\rho^{\mathsf{yes}} = \left\{\mathsf{x} \mid \exists \mathsf{w} \text{ s.t. } (\mathsf{x}, \mathsf{w}) \in \mathcal{R}_\rho^{\mathsf{yes}}\right\}, \qquad \mathcal{L}_{\rho,S}^{\mathsf{no}} = \left\{(\mathsf{x}, \mathsf{w}^*) \mid \exists \mathsf{y} \text{ s.t. } (\mathsf{x}, \mathsf{y}, \mathsf{w}^*) \in \mathcal{R}_{\rho,S}^{\mathsf{no}}\right\}$$

where w^* is the promise of the adversary and y is the local f-witness that we can extract from the adversary. Intuitively, the two witnesses of the languages are different kind of objects. Witness y is the value we extract from the adversary, which can't be equal to $f(\mathsf{w}, S)$ for successful adversaries, but should lie the image of f anyway. On the other hand w is a "proper" witness from which an y can be computed and hence belongs to the pre-image of f.[13]

[12] In the case x is a 3-CNF formula, in [KPY19] the authors define $\mathcal{R}_{\rho,S}^{\mathsf{yes}}$ as the pairs (x, w) where w is a "locally satisfying assignment". This means that every clause C in x with all variables in S, is satisfied by w.

[13] The original definition from [GR19] is syntactically different as x is part of the statement in the yes language. However, as the authors said, the verifier can't read w as it will render the verification process not succinct. Since y becomes irrelevant, we prefer to eliminate it from the yes language.

4.4.1 Arguments with No-signaling extraction and Oblivious SRS Generation

Similarly to the way we treated commitment schemes, we don't directly prove the existence of no-signaling extractors but first show the existence of an Oblivious srs Generation algorithm. We then show the latter notion implies the former. For convenience, we start defining a quasi argument without no-signaling extraction but only local knowledge soundness.

Note that in contrast with the strong soundness notion of [JR13], we do not allow the adversary to access the witness of the language parameters θ, since in some cases (knowledge transfer) soundness makes computational assumption about the values in it. As in the commitment space, we consider in the definitions subsets of $[n]$ of size bounded by K, but it straightforward to modify the definitions for more general subset families $\mathcal{S} \subseteq 2^{[n]}$.

Definition 29. An *locally extractable proof system* Π for the parameter language $\mathcal{L}_{\mathsf{par}}$ and relations $\mathcal{R}_{\rho}^{\mathsf{yes}}, \mathcal{R}_{\rho,S}^{\mathsf{no}}$ is a tuple of PPT algorithms $\Pi = (\mathsf{K}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Extract})$ where

- $(\rho, \theta) \leftarrow \mathcal{D}_{\mathsf{par}}(\mathsf{gk}, d, K, S)$: Parameter generation $\mathcal{D}_{\mathsf{par}}$ takes as input a group key gk, the locality parameter K and a set $S \subseteq ([d], \ldots, [d])$ with $|S| \leq K$; it outputs an instance witness pair (ρ, θ) of $\mathcal{L}_{\mathsf{par}}$.
- $(\mathsf{srs}, \tau) \leftarrow \mathsf{K}(\rho, \theta)$: K takes as input an instance-witness pair (ρ, θ) of $\mathcal{L}_{\mathsf{par}}$; it outputs a common reference string srs and an extraction trapdoor τ.
- $\pi \leftarrow \mathsf{Prove}(\mathsf{srs}, \mathsf{x}, \mathsf{w})$: Prove takes as input srs and a statement-witness pair (x, w) of $\mathcal{L}_{\rho}^{\mathsf{yes}}$; it outputs a proof π.
- $b \leftarrow \mathsf{Verify}(\mathsf{srs}, \mathsf{x}, \pi)$: Verify takes as input srs, a statement x and a proof π; it outputs a bit b indicating if the proof π is a valid proof.
- $y \leftarrow \mathsf{Extract}(\tau, \mathsf{x}, \pi)$: Extract takes as input the extraction trapdoor τ, a statement x and a proof π, and outputs a local witness y for the set S.

such that for all $\kappa \in \mathbb{N}, K \leq d \in \mathbb{N}, S \subseteq [d]$, with $|S| \leq K$, Π satisfies the following properties:

- Completeness: For all $(\rho, \theta) \in \mathcal{L}_{\mathsf{par}}$ and $\mathsf{x}, \mathsf{w} \in \{0,1\}^*$

$$\Pr_{\mathsf{gk} \leftarrow \mathcal{G}(1^\kappa)} \left[\begin{array}{c} \mathsf{Verify}(\mathsf{srs}, \mathsf{x}, \pi) = 1 \\ \vee\ (\mathsf{x}, \mathsf{w}) \notin \mathcal{R}_{\rho,S}^{\mathsf{yes}} \end{array} \middle| \begin{array}{c} (\mathsf{srs}, \tau) \leftarrow \mathsf{K}(\rho, \theta) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{srs}, \mathsf{x}, \mathsf{w}) \end{array} \right] \geq 1 - \mathsf{negl}(\kappa)$$

- Local Knowledge Soundness: For all PPT $\mathcal{A}$

$$\Pr_{gk \leftarrow \mathcal{G}(1^\kappa)} \left[\begin{array}{c} \mathsf{Verify}(\mathsf{srs}, \mathsf{x}, \pi) = 0 \\ \vee\ (\mathsf{x}, \mathbf{y}, \mathsf{w}^*) \notin \mathcal{R}_{\rho,S}^{\mathsf{no}} \end{array} \middle| \begin{array}{r} (\rho, \theta) \leftarrow \mathcal{D}_{\mathsf{par}}(\mathsf{gk}, d, K, S); \\ (\mathsf{srs}, \tau) \leftarrow \mathsf{K}(\rho, \theta); \\ (\mathsf{x}, \mathsf{w}^*, \pi) \leftarrow \mathcal{A}(\rho, \mathsf{srs}); \\ y \leftarrow \mathsf{Extract}(\tau, \mathsf{x}, \pi) \end{array} \right] \geq 1 - \mathsf{negl}(\kappa)$$

Next, we define the no-signaling property of quasi-arguments.

Definition 30. An locally extractable proof system Π for the parameter language $\mathcal{L}_{\mathsf{par}}$ and relations $\mathcal{R}^{\mathsf{yes}}_{\rho,S}, \mathcal{R}^{\mathsf{no}}_{\rho,S}$ is a *quasi argument* if it satisfies *no-signaling extraction*. That is, for all $\kappa \in \mathbb{N}, K \leq d, S' \subseteq S \subseteq [d]$ with $|S| \leq K$, and all PPT $\mathcal{A}$ and PPT $\mathcal{D}$

$$\left| \Pr_{\mathsf{gk} \leftarrow \mathcal{G}(1^\kappa)} \left[\mathcal{D}(\mathsf{srs}, \mathsf{x}, \pi, \mathsf{y}_{S'}) = 1 \;\middle|\; \begin{array}{r} (\rho, \theta) \leftarrow \mathcal{D}_{\mathsf{par}}(\mathsf{gk}, d, K, S) \\ (\mathsf{srs}, \tau) \leftarrow \mathsf{K}(\rho, \theta) \\ (x, \pi) \leftarrow \mathcal{A}(\rho, \mathsf{srs}) \\ \text{if } \mathsf{Verify}(\mathsf{srs}, \mathsf{x}, \pi) = 0\text{: set } \mathsf{x} = \perp \\ y \leftarrow \mathsf{Extract}(\tau, \mathsf{x}, \pi) \end{array} \right] - \right.$$
$$\left. \Pr_{gk \leftarrow \mathcal{G}(1^\kappa)} \left[\mathcal{D}(\mathsf{srs}, \mathsf{x}, \pi, \mathsf{y}') = 1 \;\middle|\; \begin{array}{r} (\rho, \theta) \leftarrow \mathcal{D}_{\mathsf{par}}(\mathsf{gk}, d, K, S') \\ (\mathsf{srs}, \tau) \leftarrow \mathsf{K}(\rho, \theta); \\ (x, \pi) \leftarrow \mathcal{A}(\rho, \mathsf{srs}) \\ \text{if } \mathsf{Verify}(\mathsf{srs}, \mathsf{x}, \pi) = 0\text{: set } \mathsf{x} = \perp; \\ \mathsf{y}' \leftarrow \mathsf{Extract}(\tau, \mathsf{x}, \pi) \end{array} \right] \right| \leq \mathsf{negl}(\kappa)$$

Finally, we define the notion of oblivious locally extractable proof systems. The requirements are that (1) the srs alone does not help PPT adversaries gain information about the extraction set used to sample the parameters ρ; (2) there exists a PPT algorithm OblSetup that on input a set $S' \subseteq S$ and (ρ, srs), sampled for extraction on the superset of S, outputs new values (ρ', srs') that are statistically close to (ρ, srs) and additionally, it outputs a trapdoor τ' for S' that outputs indistinguishable witnesses to the ones output for S and restricted to S'.

Definition 31. A locally extractable proof system Π for the parameter language $\mathcal{L}_{\mathsf{par}}$ and relations $\mathcal{R}^{\mathsf{yes}}_{\rho}, \mathcal{R}^{\mathsf{no}}_{\rho,S}$ is *Oblivious* if there exist a PPT algorithm OblSetup such that, for all $\kappa \in \mathbb{N}, K \leq d \in \mathbb{N}, S', S \subseteq [d]$ with $|S'|, |S| \leq K$,

1. *Index Set Hiding*: for all PPT $\mathcal{D}$

$$\left| \Pr_{\mathsf{gk} \leftarrow \mathcal{G}(1^\kappa)} \left[\mathcal{D}(\rho, \mathsf{srs}) \;\middle|\; \begin{array}{r} (\rho, \theta) \leftarrow \mathcal{D}_{\mathsf{par}}(\mathsf{gk}, d, K, S) \\ (\mathsf{srs}, \tau) \leftarrow \mathsf{K}(\rho, \theta) \end{array} \right] - \right.$$
$$\left. \Pr_{\mathsf{gk} \leftarrow \mathcal{G}(1^\kappa)} \left[\mathcal{D}(\rho, \mathsf{srs}) \;\middle|\; \begin{array}{r} (\rho, \theta) \leftarrow \mathcal{D}_{\mathsf{par}}(\mathsf{gk}, d, K, S') \\ (\mathsf{srs}, \tau) \leftarrow \mathsf{K}(\rho, \theta) \end{array} \right] \right| \leq \mathsf{negl}(\kappa)$$

2. *Oblivious Trapdoor Generation:* if $S' \subseteq S$ then for all, (even unbounded) adversaries

$\mathcal{A}$ and distinguishers $\mathcal{D}$

$$\left| \Pr_{\mathsf{gk}\leftarrow\mathcal{G}(1^\kappa)} \left[\mathcal{D}(\rho', \mathsf{srs}', \mathsf{y}') = 1 \;\middle|\; \begin{array}{r} (\rho,\theta) \leftarrow \mathcal{D}_{\mathsf{par}}(\mathsf{gk}, d, K, S) \\ (\mathsf{srs},\tau) \leftarrow \mathsf{K}(\rho,\theta) \\ (\rho', \mathsf{srs}', \tau') \leftarrow \mathsf{OblSetup}(\rho, \mathsf{srs}, \mathsf{S}') \\ (\mathsf{x},\pi) \leftarrow \mathcal{A}(\rho', \mathsf{srs}') \\ \text{if } \mathsf{Verify}(\mathsf{srs}', \mathsf{x}, \pi) = 0\text{: } \mathsf{x} = \bot \\ \mathsf{y}' \leftarrow \mathsf{Extract}(\tau', \mathsf{x}, \pi) \end{array} \right] - \right.$$

$$\left. \Pr_{\mathsf{gk}\leftarrow\mathcal{G}(1^\kappa)} \left[\mathcal{D}(\rho, \mathsf{srs}, \mathsf{y}_{\mathbf{S}'}) = 1) = 1 \;\middle|\; \begin{array}{r} (\rho,\theta) \leftarrow \mathcal{D}_{\mathsf{par}}(\mathsf{gk}, d, K, S) \\ (\mathsf{srs},\tau) \leftarrow \mathsf{K}(\rho,\theta) \\ (\mathsf{x},\pi) \leftarrow \mathcal{A}(\rho, \mathsf{srs}) \\ \text{if } \mathsf{Verify}(\mathsf{srs}, \mathsf{x}, \pi) = 0\text{: } \mathsf{x} = \bot \\ y \leftarrow \mathsf{Extract}(\tau, \mathsf{x}, \pi) \end{array} \right] \right| \leq \mathsf{negl}(\kappa)$$

Note that (2) holds against unbounded adversaries which can compute θ by themselves.

Next, we present a proof that if a locally extractable proof system satisfies oblivious srs generation, then it is no-signaling. The proof is similar to the proof of Thm. 10.

Theorem 15. *Let* $\Pi = (\mathsf{K}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Extract}, \mathsf{OblSetup})$ *be an Oblivious Locally Extractable Proof System for the parameter language* $\mathcal{L}_{\mathsf{par}}$ *and relations* $\mathcal{R}^{\mathsf{yes}}_{\rho}, \mathcal{R}^{\mathsf{no}}_{\rho,\mathbf{S}}$*. Then,* Π *has no signaling extraction.*

Proof. Fix any $S' \subseteq S \subseteq [d]$ with $|S| \leq K$, and let $\mathcal{D}$ be a PPT distinguisher against no signaling extraction for these values, on instance-proof pairs output by a PPT $\mathcal{A}$. We show by a sequence of games that its success probability is negligible.

$\mathsf{Game}_0^{\mathcal{D},\mathcal{A}}(1^\kappa)$: We execute $(\rho,\theta) \leftarrow \mathcal{D}_{\mathsf{par}}(\mathsf{gk}, d, K, S)$; $(\mathsf{srs},\tau) \leftarrow \mathsf{K}(\rho,\theta)$; we then get $(\mathsf{x},\pi) \leftarrow \mathcal{A}(\rho, \mathsf{srs})$ and change x to $\bot$ if $\mathsf{Verify}(\mathsf{srs}, \mathsf{x}, \pi) = 0$; we compute $\mathsf{y} \leftarrow \mathsf{Extract}(\tau, \mathsf{x}, \pi)$. The output is $\mathcal{D}(\mathsf{srs}, \mathsf{x}, \pi, \mathsf{y}_{S'})$.

$\mathsf{Game}_1^{\mathcal{D},\mathcal{A}}(1^\kappa)$: We execute $(\rho,\theta) \leftarrow \mathcal{D}_{\mathsf{par}}(\mathsf{gk}, d, K, S)$; $(\mathsf{srs},\tau) \leftarrow \mathsf{K}(\rho,\theta)$; we use the oblivious extractor to get $(\rho', \mathsf{srs}', \tau') \leftarrow \mathsf{OblSetup}(\rho, \mathsf{srs}, \mathsf{S}')$; we then get $(\mathsf{x},\pi) \leftarrow \mathcal{A}(\rho', \mathsf{srs}')$ and change x to $\bot$ if $\mathsf{Verify}(\mathsf{srs}, \mathsf{x}, \pi) = 0$; we compute $y' \leftarrow \mathsf{Extract}(\tau', \mathsf{x}, \pi)$. The output is $\mathcal{D}(\mathsf{srs}', \mathsf{x}, \pi, \mathsf{y}')$.

$\mathsf{Game}_2^{\mathcal{D},\mathcal{A}}(1^\kappa)$: This is the same as $\mathsf{Game}_1^{\mathcal{D},\mathcal{A}}$ but in the first step we sample parameters for S', that is we execute $(\rho,\theta) \leftarrow \mathcal{D}_{\mathsf{par}}(\mathsf{gk}, d, K, S')$.

$\mathsf{Game}_3^{\mathcal{D},\mathcal{A}}(1^\kappa)$: We execute $(\rho,\theta) \leftarrow \mathcal{D}_{\mathsf{par}}(\mathsf{gk}, d, K, S')$; $(\mathsf{srs},\tau) \leftarrow \mathsf{K}(\rho,\theta)$; we then get $(\mathsf{x},\pi) \leftarrow \mathcal{A}(\rho, \mathsf{srs})$ and change x to $\bot$ if $\mathsf{Verify}(\mathsf{srs}, \mathsf{x}, \pi) = 0$; we compute $\mathsf{y}' \leftarrow \mathsf{Extract}(\tau, \mathsf{x}, \pi)$. The output is $\mathcal{D}(\mathsf{srs}, \mathsf{x}, \pi, \mathsf{y}')$.

We next show that for all $1 \leq i \leq 3$,

$$\left|\Pr\left[\mathsf{Game}_i^{\mathcal{D},\mathcal{A}}(1^\kappa) = 1\right] - \Pr\left[\mathsf{Game}_{i-1}^{\mathcal{D},\mathcal{A}}(1^\kappa) = 1\right]\right| \leq \mathsf{negl}(\kappa) \tag{4.4}$$

- *Case* $i = 1$, $i = 3$. Note that for $i = 1$, the difference in the two games is exactly as in the two cases of the oblivious trapdoor generation property for $S' \subseteq S$, so the outputs of games are statistically close. For case 3, we use the same argument for $S' \subseteq S'$.
- *Case* $i = 2$ The only difference in the games is how we setup the initial srs, either by sampling for S' or for S. The output of the two games are computationally indistinguishable by the index set hiding property, even when the adversary is given $h_{\mathsf{ns}}(\theta)$.

By a standard argument we get that, for all PPT $\mathcal{D}, \mathcal{A}$,

$$\left|\Pr\left[\mathsf{Game}_0^{\mathcal{D},\mathcal{A}}(1^\kappa) = 1\right] - \Pr\left[\mathsf{Game}_5^{\mathcal{D},\mathcal{A}}(1^\kappa) = 1\right]\right| \leq \mathsf{negl}(\kappa)$$

Finally, noting that $\mathsf{Game}_0^{\mathcal{D},\mathcal{A}}$, $\mathsf{Game}_3^{\mathcal{D},\mathcal{A}}$ correspond to the two cases of no signaling extraction, we conclude the proof. □

4.4.2 Succinct Pairing Based Quasi-Arguments

In this section we present quasi arguments for various languages using SSB commitments with oblivious trapdoor generation. We first present the simpler case, membership in linear spaces, and then we present some extensions of it, specifically a knowledge transfer version, and a knowledge transfer version for statements split in the two groups. Finally, we use the latter to build a quasi argument of knowledge transfer for Hadamard products.

In this section, we only present a detailed security analysis for the former construction; the rest are deferred to Sec. 4.7. This is because the analysis of the latter constructions essentially consist of slightly "tweaking" the linear space membership construction. Thus, in this section, we only present the statements of the properties they satisfy in the context on which we will use them in the delegation construction.

4.4.2.1 Quasi Arguments of Membership in Linear Spaces

Let $\mathcal{U}$ be a witness samplable distributions sampling $([\mathbf{U}]_1, \mathbf{U})$, where $\mathbf{U} \in \mathbb{F}^{d\times n}$. We assume that for any $S \subseteq [d]$, given only $[\mathbf{U}]_1$ such that $\mathbf{U} = \mathbf{P}_S^\top \begin{pmatrix} \mathbf{U}_S \\ \mathbf{U}_{\overline{S}} \end{pmatrix}$ there exists an efficient way of sampling $[\mathbf{U}_{\overline{S}}]$.[14] Also, let CS be an algebraic SSB commitment key. The parameter language

[14] We will instantiate the argument with $\mathbf{U}$ a block lower triangular matrix where each row is of the form $(\mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_i, \mathbf{0}, \ldots, \mathbf{0})$ where $\{\mathbf{U}_i\}_i$ are independent random variables. Then is clear that from $[\mathbf{U}_S]_1$ we know $[\mathbf{U}_i]_1$ up to $i = \max S$, and the rest $\{\mathbf{U}_j : j \notin S\}$ can be sampled independently.

Figure 4.5 Construction QALin for membership in linear spaces. Note that this is just the argument of [KW15] for matrix $[\mathbf{GU}]_1$.

$\mathcal{D}_{\mathsf{par}}(\mathsf{gk}, d, K, S)$**:**

$([\mathbf{U}]_1, \mathbf{U}) \leftarrow \mathcal{U}$

$([\mathbf{G}]_1, \mathbf{T}, \mathbf{G}) \leftarrow \mathsf{CS.KeyGen}(\mathsf{gk}, d, K, S)$

Output (ρ, θ) where $\rho = (\mathsf{gk}, [\mathbf{G}]_1, [\mathbf{U}]_1), \theta = (\mathbf{G}, \mathbf{U}, \mathbf{T})$

$\mathsf{K}(\rho, \theta)$**:**

Sample $\mathbf{K} \leftarrow \mathbb{F}^{K+k \times k}$, $\mathbf{A} \leftarrow \mathcal{D}_k$, and redefine $\mathbf{A}$ as its first k columns

Compute $[\mathbf{B}]_1 = [\mathbf{U}^\top]_1 \mathbf{G}^\top \mathbf{K}$, $\mathbf{C} = \mathbf{KA}$

Output (srs, τ) where $\mathsf{srs} = ([\mathbf{A}]_2, [\mathbf{B}]_1, [\mathbf{C}]_2)$, $\tau = \mathbf{T}$

$\mathsf{Prove}(\mathsf{srs}, [\mathbf{c}]_1, \mathbf{w})$**:**

Output $[\boldsymbol{\pi}]_1 \leftarrow \mathbf{w}^\top [\mathbf{B}]_1$

$\mathsf{Verify}(\mathsf{srs}, [\mathbf{c}]_1, [\boldsymbol{\pi}]_1)$**:**

Output 1 iff $e([\boldsymbol{\pi}]_1, [\mathbf{A}]_2) = e([\mathbf{c}]_1, [\mathbf{C}]_2)$ and 0

$\mathsf{Extract}(\tau, [\mathbf{c}]_1, [\boldsymbol{\pi}]_1)$**:**

Output $[\mathbf{y}]_1 \leftarrow \mathbf{T}^\top [\mathbf{c}_S]_1$, otherwise output $\perp$

is

$$\mathcal{L}_{\mathsf{par}} = \{[\mathbf{U}]_1, [\mathbf{G}]_1 \mid \exists \mathbf{U}, \mathbf{G} \text{ s.t. } ([\mathbf{U}]_1, \mathbf{U}) \in \mathsf{Sup}(\mathcal{U}) \text{ and } ([\mathbf{G}]_1, \mathbf{G}, \mathbf{T}) \in \mathsf{Sup}(\mathsf{CS.KeyGen}(\mathsf{gk}, d, K, S))\}$$

We assume that the corresponding relation is efficiently verifiable[15]. The parameters $\rho = ([\mathbf{U}]_1, [\mathbf{G}]_1) \leftarrow (\mathcal{U}, \mathsf{CS.KeyGen}(\mathsf{gk}, d, K, S))$ define the following relations:

$$\mathsf{Lin}\text{-}\mathcal{R}^{\mathsf{yes}}_{\rho} = \{([\mathbf{c}]_1, \mathbf{w}) : \mathbf{c} = \mathbf{GUw}\},$$
$$\mathsf{Lin}\text{-}\mathcal{R}^{\mathsf{no}}_{\rho,S} = \{([\mathbf{c}]_1, [\mathbf{y}]_1) : \mathbf{y} \text{ is a valid } S\text{-opening of } \mathbf{c} \text{ and } \mathbf{y} \notin \mathrm{Im}(\mathbf{U}_S)\}$$

The advice is the empty string while the extractor should retrieve $f(\mathbf{w}, S) = [\mathbf{U}_A]_1 \mathbf{w}$ from any accepting statement and proof pair. We present the construction QALin in Fig. 4.5. The construction is essentially the quasi adaptive construction of membership in linear space of [KW15] for the matrix $[\mathbf{GU}]_1$.

Remark (Knowledge transfer variant)**.** Following the analysis of [GR19], one can also consider a variant of the construction that satisfies knowledge transfer, assuming some

[15]This is w.l.o.g. since one can extend the witness to include the randomness used to sample the parameters.

hardness assumption (MDDH) for the distribution $\mathcal{U}$. We omit presenting it since we will present a more general variant, knowledge transfer for the bilateral case.

We next show that the construction is a quasi argument of knowledge. We first consider the local extractability properties of the construction and then the oblivious trapdoor generation, which implies the no-signaling property.

First, we prove local soundness. Our strategy is to reduce the local knowledge soundness to the soundness of the membership in linear spaces construction of [KW15] for the sum-matrix of $\mathbf{U}$ defined by S. Recall that the soundness of the latter construction holds under the $\mathcal{D}_k$-KerMDH assumption.

Theorem 16. *Let $\mathcal{U}$ be a witness samplable distribution, $\mathcal{D}_k$ be a matrix distribution for which $\mathcal{D}_k$*KerMDH *holds, and* CS *an algebraic SSB commitment. Then, construction* QALin *of Fig. 4.5 satisfies local knowledge soundness for* (Lin-$\mathcal{R}_\rho^{\mathsf{yes}}$, Lin-$\mathcal{R}_{\rho,S}^{\mathsf{no}}$).

Proof. For completeness, we have that if $\mathbf{c} = \mathbf{GUw}$, then

$$\mathbf{c}^\top \mathbf{C} = (\mathbf{GUw})^\top \mathbf{C} = \mathbf{w}^\top \mathbf{U}^\top \mathbf{G}^\top \mathbf{C} = \mathbf{w}^\top \mathbf{U}^\top \mathbf{G}^\top \mathbf{KA} = \mathbf{w}^\top \mathbf{BA} = \boldsymbol{\pi}\mathbf{A}.$$

Local knowledge soundness is guaranteed by the local extractability of the SSB commitment scheme and soundness of Kiltz and Wee proof system. Note that the extractor always outputs a valid partial opening of $[\mathbf{c}]_1$ given an accepting proof $[\boldsymbol{\pi}]_1$, by the local extractability property of the SSB commitments. We claim that this opening must lie in $\mathrm{Im}([\mathbf{U}_S]_1)$. Assume otherwise, and let $\mathcal{A}$ be a PPT adversary that makes the extraction fail. We construct a PPT adversary $\mathcal{B}_S$ that breaks strong soundness of Kiltz and Wee for the matrix $\mathbf{U}_S$, conditioned on $\mathcal{A}$ giving a valid proof. $\mathcal{B}_S$ works as follows: it takes input srs_S containing $[\mathbf{U}_S]_1 \in \mathbb{G}^{|S|\times d}$, $[\mathbf{A}]_2 \in \mathbb{G}_2^{k\times k}$, $[\mathbf{B}^\dagger]_1 \in \mathbb{G}^{d\times k}$, $[\mathbf{C}^\dagger]_2 \in \mathbb{G}_2^{|S|\times k}$ and the discrete logarithms of matrix $\mathbf{U}_S$ and does the following:

- It samples $([\mathbf{U}_{\overline{S}}]_1, \mathbf{U}_{\overline{S}})$ s.t. $\mathbf{U} = \mathbf{P}_S^\top \begin{pmatrix} \mathbf{U}_S \\ \mathbf{U}_{\overline{S}} \end{pmatrix}$.
- It samples $([\mathbf{G}]_1, \mathbf{G}, \mathbf{T}) \leftarrow \mathsf{CS.KeyGen}(\mathsf{gk}, n, d, K, S)$ and a random matrix $\mathbf{R} \leftarrow \mathbb{F}^{K+k\times k}$.
- It computes $[\mathbf{B}]_1 = [\mathbf{B}^\dagger]_1 + [\mathbf{U}]_1^\top \mathbf{G}^\top \mathbf{R}$, $[\mathbf{C}]_2 = \mathbf{T}[\mathbf{C}^\dagger]_2 + \mathbf{R}[\mathbf{A}]_2$.
- It sets $\rho := (\mathsf{gk}, [\mathbf{G}]_1, [\mathbf{U}]_1)$ and $\mathsf{srs} := ([\mathbf{A}]_2, [\mathbf{B}]_1, [\mathbf{C}]_2)$.

It then executes $\mathcal{A}(\rho, \mathsf{srs})$ until it outputs $[\mathbf{c}]_1, [\boldsymbol{\pi}]_1$. If this is an accepting proof pair, $\mathcal{B}_S$ sets $[\mathbf{x}^\dagger]_1 := \mathbf{T}[\mathbf{c}]_1$ and $[\boldsymbol{\pi}^\dagger]_1 := [\boldsymbol{\pi}]_1 - [\mathbf{c}]_1^\top \mathbf{R}$.

First, we claim that the values ρ, srs given as input to $\mathcal{A}$ are identically distributed to honestly created ones and thus do not skew the probability that $\mathcal{A}$ outputs a valid proof. This is immediate for ρ since it is sampled honestly. We show that this is true for srs as well.

Let $\mathbf{K}^\dagger \in \mathbb{F}^{|S|\times k}$ be the implicit matrix in srs_S, that is, it satisfies $\mathbf{B}^\dagger = \mathbf{U}_S^\top \mathbf{K}^\dagger$ and $\mathbf{C}^\dagger = \mathbf{K}^\dagger \mathbf{A}$. Consider the matrix $\mathbf{K} = \mathbf{T}\mathbf{K}^\dagger + \mathbf{R}$, and note that this matrix is uniformly distributed since $\mathbf{R}$ is uniformly distributed. Thus $\mathbf{K}$ is distributed identically to an honestly generated $\mathbf{K}'$ for generating a srs. We claim that the srs output by $\mathcal{B}_S$ is identically distributed to sampling this matrix and computing the other values honestly. Indeed we have that

$$\begin{aligned} \mathbf{C} &= \mathbf{T}\mathbf{C}^\dagger + \mathbf{R}\mathbf{A} \\ &= \mathbf{T}\mathbf{K}^\dagger\mathbf{A} + \mathbf{R}\mathbf{A} \\ &= (\mathbf{T}\mathbf{K}^\dagger + \mathbf{R})\mathbf{A} \\ &= \mathbf{K}\mathbf{A} \end{aligned} \qquad \text{and} \qquad \begin{aligned} \mathbf{B} &= \mathbf{B}^\dagger + \mathbf{U}^\top\mathbf{G}^\top\mathbf{R} \\ &= \mathbf{U}_S^\top\mathbf{K}^\dagger + \mathbf{U}^\top\mathbf{G}^\top\mathbf{R} \\ &= \mathbf{U}^\top\mathbf{G}^\top\mathbf{T}\mathbf{K}^\dagger + \mathbf{U}^\top\mathbf{G}^\top\mathbf{R} \\ &= \mathbf{U}^\top\mathbf{G}^\top(\mathbf{T}\mathbf{K}^\dagger + \mathbf{R}) \\ &= (\mathbf{G}\mathbf{U})^\top\mathbf{K} \end{aligned}$$

where the second equality for $\mathbf{B}$ follows since by the properties of algebraic SSB commitments we have $\mathbf{T}^\top\mathbf{G} = \begin{pmatrix}\mathbf{I}_{|S|} & \mathbf{0}\end{pmatrix}\mathbf{P}_S$ which gives

$$\mathbf{U}^\top\mathbf{G}^\top\mathbf{T} = \mathbf{U}^\top\mathbf{P}_S^\top\begin{pmatrix}\mathbf{I}_{|S|}\\ \mathbf{0}\end{pmatrix} = \mathbf{U}_S$$

So, the outputted srs srs′ is indeed identically distributed with an honest one.

Finally, we show that if $\mathcal{A}$ outputs a valid proof $[\boldsymbol{\pi}]_1$, then $\mathcal{B}_S$ outputs a valid statement-proof pair w.r.t. to srs_S. Indeed, by the local extractability property of the commitment scheme, $\mathcal{B}_S$ always outputs some $[\mathbf{x}^\dagger]_1$ consistent with $[\mathbf{c}]_1$, and also the proof verifies, since we have

$$\boldsymbol{\pi}\mathbf{A} = \mathbf{c}^\top\mathbf{C} = \mathbf{c}^\top\mathbf{K}\mathbf{A} = \mathbf{c}^\top(\mathbf{T}\mathbf{K}^\dagger + \mathbf{R})\mathbf{A} = (\mathbf{x}^\dagger)^\top\mathbf{K}^\dagger\mathbf{A} + \mathbf{c}^\top\mathbf{R}\mathbf{A}$$

which gives $\boldsymbol{\pi}^\dagger\mathbf{A} = \boldsymbol{\pi}\mathbf{A} - \mathbf{c}^\top\mathbf{R}\mathbf{A} = (\mathbf{x}^\dagger)^\top\mathbf{K}^\dagger\mathbf{A} = (\mathbf{x}^\dagger)\mathbf{C}^\dagger$. We conclude that $[\boldsymbol{\pi}^\dagger]_1$ is a valid proof for $[\mathbf{x}^\dagger]_1 \notin \mathrm{Im}([\mathbf{U}_S]_1)$ and $\mathcal{B}_S$ breaks soundness of Kiltz and Wee construction. □

We next show that the QALin construction is no-signaling. The property relies on the oblivious trapdoor generation and index set hiding of SSB commitments. We note that the property holds even if the adversary knows the discrete logarithms of elements in $\mathbf{U}$ as well as any information about the commitment key that does not break the index set hiding property.

Our proof strategy is the following: first, we show that there is an alternative way to sample the srs and then we use this alternative key generation algorithm in proving that QALin is oblivious. Then, the no-signaling property follows by Thm. 15.

Lemma 1. *There exists a modified* srs *generation algorithm* K' *that on input* $(\rho, \mathbf{U})$ *outputs an* srs *such that* (ρ, srs) *are identically distributed to the output of the honest algorithm* $\mathsf{K}(\rho, \theta)$.

The lemma follows directly by noting that $[\mathbf{B}]_1 = [\mathbf{U}^\top]_1\mathbf{G}\mathbf{K} = \mathbf{U}^\top[\mathbf{G}]_1\mathbf{K}$. Given that this result holds, we slightly abuse notation and refer to K' as K, that is we use the same name for the honest and the simulated algorithm.

Theorem 17. *Let $\mathcal{U}$ be a witness samplable distribution, and* CS *be an algebraic SSB commitment scheme with perfect completeness, index set hiding and oblivious trapdoor generation. Then Construction* QALin *of Fig. 4.5 is oblivious.*

Proof. For index set hiding, it is enough to notice that in both cases, the srs of QALin can be efficiently computed given only $\mathsf{ck} = [\mathbf{G}]_1$. Indeed, by sampling $[\mathbf{U}]_1, \mathbf{U} \leftarrow \mathcal{U}$, all values of srs are efficiently computable, as noted in the Lemma 1. Thus, a distinguishing advantage in index set hiding of QALin immediately implies equal advantage on the index set hiding property of CS.

For oblivious trapdoor generation we first describe the OblSetup algorithm. Let $S' \subseteq S$.

$\mathsf{OblSetup}(\rho = ([\mathbf{G}]_1, [\mathbf{U}]_1), \mathsf{srs})$**:**

$([\mathbf{G}']_1, \mathbf{T}') \leftarrow \mathsf{CS.OblSetup}(\mathsf{gk}, \mathsf{d}, \mathsf{K}, \mathsf{S}, \mathsf{ck} = [\mathbf{G}]_1)$
$([\mathbf{U}]_1, \mathbf{U}) \leftarrow \mathcal{U}$
$(\mathsf{srs}, \tau) \leftarrow \mathsf{QALin.K}(([\mathbf{G}']_1, [\mathbf{U}]_1), \mathbf{U})$

Note that the only difference in sampling with S and with S' is how we sample the commitment key $\mathbf{G}$. The srs part srs is identically distributed to an honest one by Lemma 1. Finally, by the statistically binding property of the commitment key the extracted witness for S and S' are unique and thus do not help the (unbounded) distinguisher, who can compute them on its own. □

We summarize the properties of the construction in the next theorem.

Theorem 18. *Let $\mathcal{D}_k$ be a matrix distribution for which $\mathcal{D}_k$-*KerMDH *assumption holds, and* CS *be an algebraic SSB commitment scheme with index set hiding and oblivious extractability. Then construction* QALin *is a quasi argument.*

Proof. The local extractability property follows from Thm 16 and the no-signaling property follows by Thm. 15 and the oblivious trapdoor generation property of QALin, which in turn follows from Thm 17. □

We next consider extensions of QALin:

1. a knowledge transfer variant for bilateral linear spaces [GHR15], where the statement as well as the generating matrix have components in both groups.
2. a knowledge transfer "sum" argument [GHR15] which is akin to a bilateral language but one shows that the sum of the discrete logs of two vectors in $\mathbb{G}_1$ and $\mathbb{G}_2$ belong to the image of the sum of two matrices in $\mathbb{G}_1, \mathbb{G}_2$.

The security analysis of both extensions is almost verbatim to the analysis of QALin. One difference is that in these cases, we rely on soundness of the knowledge transfer variant of [KW15] which is sound under standard assumptions [GR19], so we manage to achieve these properties under falsifiable and standard assumptions. The security analysis is almost identical to that of QALin, so we simply state the properties and we defer it to Sec. 4.7.1.

Quasi Argument for Bilateral Linear Knowledge Transfer. Consider three $d \times n$ matrices $[\mathbf{M}]_1$, $[\mathbf{N}_1]_1$, $[\mathbf{N}_2]_2$, a vector $\mathbf{w} \in \mathbb{F}^n$ and vectors $[\mathbf{x}]_1$, $[\mathbf{y}_1]_1$, $[\mathbf{y}_2]_1$. Now, if $S_1, S_2 \subseteq [d]$, we want to consider local constraints of the form:

$$\text{“if } [\mathbf{x}]_1 = [\mathbf{M}_{S_1}]_1\mathbf{w} \text{ then } [\mathbf{y}_1]_1 = [\mathbf{N}_{1,S_2}]_1\mathbf{w} \text{ and } [\mathbf{y}_2]_1 = [\mathbf{N}_{2,S_2}]_1\mathbf{w}\text{”}.$$

For efficiency reasons, we want to avoid sending the whole vectors

$$[\mathbf{x}]_1 = [\mathbf{M}]_1\mathbf{w}, \qquad [\mathbf{y}_1]_1 = [\mathbf{N}_1]_1\mathbf{w}, \qquad [\mathbf{y}_2]_2 = [\mathbf{N}_2]_2\mathbf{w}$$

Instead, we commit to these elements using three different (algebraic) SSB commitments $[\mathbf{G}]_1, [\mathbf{H}]_1, [\mathbf{F}]_1$, which we set extractable at S_1, S_2, S_2 respectively. Our ultimate goal is to receive (succinct) commitments

$$[\mathbf{c}]_1 = [\mathbf{GM}]_1\mathbf{w}, \qquad [\mathbf{d}_1]_1 = [\mathbf{HN}_1]_1\mathbf{w}, \qquad [\mathbf{d}_2]_2 = [\mathbf{FN}_2]_2\mathbf{w}$$

extract them at S_1, S_2, S_2 respectively, and make sure the above knowledge transfer notion is satisfied for the extracted values and a claim $\mathbf{w}$. Note that in this case, we do not consider simple constraint subsets as in the construction QALin. Rather, our constraints can be described as the pair of sets $S_1, S_2 \subseteq [d]$ each of size at most K_1, K_2 respectively. We extract the first commitment in the position defined by S_1, and the other two by the positions defined by S_2. We next formalize the above discussion and present the construction.

Let $\mathcal{M}, \mathcal{N}_1, \mathcal{N}_2$ be three witness samplable distribution over matrices in $\mathbb{G}^{d\times n}$, $\mathbb{G}^{d\times n}$ and $\mathbb{G}^{d\times n}$ respectively, for $n, d \in \mathbb{N}$. Let $\mathbf{K} = (K_1, K_2)$ with $K_i \leq d$ and $\mathbf{S} = (S_1, S_2)$ with $|S_i| \subseteq [d]$. Also, let CS be an algebraic SSB commitment schemes with commitment space $\mathbb{G}^{\overline{K}}$ for $\overline{K} \geq \max(K_1, K_2)$, where $\mathbb{G}$ is defined by the group key gk. The parameter language is

$$\begin{aligned}\mathcal{L}_{\mathsf{par}} = \big\{&[\mathbf{M}]_1,[\mathbf{N}_1]_1, [\mathbf{N}_2]_2, [\mathbf{G}]_1, [\mathbf{H}]_1, [\mathbf{F}]_2 \mid \exists \mathbf{M}, \mathbf{N}_1, \mathbf{N}_2, \mathbf{G}, \mathbf{H}, \mathbf{F} \text{ s.t.}\\ &([\mathbf{M}]_1, \mathbf{M}), ([\mathbf{N}_1]_1, \mathbf{N}_2), ([\mathbf{N}_2]_2, \mathbf{N}_2) \in \mathsf{Sup}(\mathcal{M}, \mathcal{N}_1, \mathcal{N}_2),\\ &([\mathbf{G}]_1, \mathbf{G}, \mathbf{T}_\mathbf{G}) \in \mathsf{Sup}(\mathsf{CS.KeyGen}(\mathsf{gk}, d, K_1, S_1)),\\ &([\mathbf{H}]_1, \mathbf{H}, \mathbf{T}_\mathbf{H}) \in \mathsf{Sup}(\mathsf{CS.KeyGen}(\mathsf{gk}, d, K_2, S_2)),\\ &([\mathbf{F}]_2, \mathbf{F}, \mathbf{T}_\mathbf{F}) \in \mathsf{Sup}(\mathsf{CS.KeyGen}(\mathsf{gk}, d, K_2, S_2))\big\}\end{aligned}$$

We assume w.l.o.g. that the corresponding relation is efficiently verifiable. The parameters $\rho = ([\mathbf{M}]_1, [\mathbf{N}_1]_1, [\mathbf{N}_2]_2, [\mathbf{G}]_1, [\mathbf{H}]_1, [\mathbf{F}]_2)$ define the following relations:

Figure 4.6 Quasi argument QABLin for knowledge transfer of membership in bilateral linear spaces.

$\mathcal{D}_{\mathsf{par}}(\mathsf{gk}, d, \mathbf{K} = (K_1, K_2), \mathbf{S} = (S_1, S_2))$**:**

$([\mathbf{M}]_1, \mathbf{M}) \leftarrow \mathcal{M}\ ([\mathbf{N}_1]_1, \mathbf{N}_1) \leftarrow \mathcal{N}_1\ ([\mathbf{N}_2]_2, \mathbf{N}_2) \leftarrow \mathcal{N}_2$

$([\mathbf{G}]_1, \mathbf{G}, \mathbf{T_G}) \leftarrow \mathsf{CS.KeyGen}(\mathsf{gk}_1, n, d, K_1, S_1)$

$([\mathbf{H}]_1, \mathbf{H}, \mathbf{T_H}) \leftarrow \mathsf{CS.KeyGen}(\mathsf{gk}_1, n, d, K_2, S_2)$

$([\mathbf{F}]_2, \mathbf{F}, \mathbf{T_F}) \leftarrow \mathsf{CS.KeyGen}(\mathsf{gk}_2, n, d, K_2, S_2)$

Output (ρ, θ) where

$$\rho = (\mathsf{gk}, [\mathbf{G}]_1, [\mathbf{H}]_1, [\mathbf{F}]_2, [\mathbf{M}]_1, [\mathbf{N}_1]_1, [\mathbf{N}_2]_2), \qquad \theta = (\mathbf{G}, \mathbf{H}, \mathbf{F}, \mathbf{T_G}, \mathbf{T_H}, \mathbf{T_F}, \mathbf{M}, \mathbf{N}_1, \mathbf{N}_2)$$

$\mathsf{K}(\rho, \theta)$**:**

Sample $\mathbf{K}_0 \leftarrow \mathbb{F}^{\overline{K}_1 \times k}$ $\mathbf{K}_1 \leftarrow \mathbb{F}^{\overline{K}_2 \times k}$ $\mathbf{K}_2 \leftarrow \mathbb{F}^{\overline{K}_2 \times k}$ $\mathbf{A} \leftarrow \mathcal{D}_k$ and redefine $\mathbf{A}$ as its first k columns

Compute $[\mathbf{B}]_1 = [\mathbf{M}^\top]_1 \mathbf{G}^\top \mathbf{K}_0 + [\mathbf{N}_1^\top]_1 \mathbf{H}^\top \mathbf{K}_1$ and $[\mathbf{D}]_2 = [\mathbf{N}_2^\top]_2 \mathbf{F}^\top \mathbf{K}_2$

$\mathbf{C}_1 = \begin{pmatrix} \mathbf{K}_0 \\ \mathbf{K}_1 \end{pmatrix} \mathbf{A}$ and $\mathbf{C}_2 = \mathbf{K}_2 \mathbf{A}$

Output (srs, τ) where $\mathsf{srs} = ([\mathbf{B}]_1, [\mathbf{D}]_2, [\mathbf{A}]_{1,2}, [\mathbf{C}_1]_2, [\mathbf{C}_2]_1)$ and $\tau = (\mathbf{T_G}, \mathbf{T_H}, \mathbf{T_F})$

$\mathsf{Prove}(\mathsf{srs}, [\mathbf{c}]_1, [\mathbf{d}_1]_1, [\mathbf{d}_2]_2, \mathbf{w})$**:**

Output $([\boldsymbol{\pi}]_1, [\boldsymbol{\theta}]_2) \leftarrow (\mathbf{w}^\top [\mathbf{B}]_1, \mathbf{w}^\top [\mathbf{D}]_2)$

$\mathsf{Verify}\ (\mathsf{srs}, ([\mathbf{c}]_1, [\mathbf{d}_1]_1, [\mathbf{d}_2]_2)\ , ([\boldsymbol{\pi}]_1, [\boldsymbol{\theta}]_2))$**:**

Output 1 iff $e\left([\boldsymbol{\pi}]_1, [\mathbf{A}]_2\right) + e\left([\boldsymbol{\theta}]_2, [\mathbf{A}]_1\right) = e\left([\mathbf{c}^\top \mid \mathbf{d}_1^\top]_1, [\mathbf{C}_1]_2\right) + e\left([\mathbf{d}_2^\top]_2, [\mathbf{C}_2]_1\right)$

$\mathsf{Extract}\ (\tau, [\mathbf{c}]_1, [\mathbf{d}_1]_1, [\mathbf{d}_2]_2, [\boldsymbol{\pi}]_1, [\boldsymbol{\theta}]_2)$**:**

Output $[\mathbf{x}]_1 = \mathbf{T}_\mathbf{G}^\top [\mathbf{c}]_1, [\mathbf{y}_1]_1 = \mathbf{T}_\mathbf{H}^\top [\mathbf{d}_1]_1, [\mathbf{y}_2]_2 = \mathbf{T}_\mathbf{F}^\top [\mathbf{d}_2]_2$

$$\mathsf{KTBLin}\text{-}\mathcal{R}_\rho^{\mathsf{yes}} = \left\{ [\mathbf{c}]_1, [\mathbf{d}_1]_1, [\mathbf{d}_2]_2, \mathbf{w} \;\middle|\; \begin{pmatrix} \mathbf{c} \\ \mathbf{d}_1 \\ \mathbf{d}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{GM} \\ \mathbf{HN}_1 \\ \mathbf{FN}_2 \end{pmatrix} \mathbf{w} \right\},$$

$$\mathsf{KTBLin}\text{-}\mathcal{R}_{\rho,\mathbf{S}}^{\mathsf{no}} = \left\{ \begin{array}{c} ([\mathbf{c}]_1, [\mathbf{d}_1]_1, [\mathbf{d}_2]_2) \\ ([\mathbf{x}]_1, [\mathbf{y}_1]_1, [\mathbf{y}_2]_2) \\ \mathbf{w} \end{array} \;\middle|\; \begin{array}{c} \mathbf{x}, \mathbf{y_1}, \mathbf{y_2} \text{ are valid } S_1, S_2, S_2 \text{ openings of} \\ \mathbf{c}, \mathbf{d}_1, \mathbf{d}_2 \text{ w.r.t. } \mathbf{G}, \mathbf{H}, \mathbf{F} \text{ respectively and} \\ \mathbf{x}_1 = \mathbf{M}_{S_1} \mathbf{w} \text{ but } \mathbf{y}_1 \neq \mathbf{N}_{1,S_2} \mathbf{w} \text{ or } \mathbf{y}_2 \neq \mathbf{N}_{2,S_2} \mathbf{w} \end{array} \right\}$$

That is, the partial witness for $\mathbf{S}$ is some valid local openings $[\mathbf{x}]_1, [\mathbf{y}_1]_1, [\mathbf{y}_2]_2$ w.r.t. to $\mathbf{G}, \mathbf{H}, \mathbf{F}$ respectively that satisfy the following: if $\mathbf{x}_{S_2} = \mathbf{M}_{S_1} \mathbf{w}$ then it should be the case that both $\mathbf{y_1} = \mathbf{N}_{1,S_2} \mathbf{w}$ and $\mathbf{y_2} = \mathbf{N}_{2,S_2} \mathbf{w}$ where $\mathbf{w}$ is the promise of the adversary. Note that if S_1 is the empty set the latter relations trivially hold. We present the protocol in Fig. 4.6.

We present a detailed security analysis of QABLin in Sec. 4.7.1. We next summarize the

properties of a specific instantiation of QABLin we will use in our delegation construction.

Theorem 19. *Construction* QABLin *instantiated with*

(1) MPed *of Fig 4.3 for the first instance of* CS

(2) source keys of KMPed *of Fig 4.4 for the second and third instance of* CS

(3) $\mathcal{M}$ *that satisfies the* $\mathcal{M}^\top$-MDDH *assumption*

(4) $\mathcal{D}_k$ *that satisfies the* $\mathcal{D}_k$-SKerMDH *assumption*

is a quasi argument. Furthermore,

- *Local knowledge soundness holds even against adversaries that know the discrete logarithms of* $\mathbf{G}, \mathbf{H}, \mathbf{F}, \mathbf{N}_1, \mathbf{N}_2$.
- *No-signaling holds even against adversaries that know the discrete logarithms of* $\mathbf{M}, \mathbf{N}_1, \mathbf{N}_2$.

Quasi Argument for Sum Knowledge Transfer. Next we consider the second variant, that Sum Knowledge Transfer construction. This will be used as a component of the Hadamard construction we will present next.

Consider three $d \times n$ matrices $[\mathbf{M}_1]_1$, $[\mathbf{N}_1]_2$, $[\mathbf{N}_2]_2$, a vector $\mathbf{w} \in \mathbb{F}^n$ and vectors $[\mathbf{x}_1]_1$, $[\mathbf{x}_2]_2$, $[\mathbf{y}]_1$. Now, if $S_0, S_1 \subseteq [d]$, we want to consider local constraints of the form $[\mathbf{x}_1]_1, [\mathbf{x}_2]_2, [\mathbf{y}]_1$ satisfy:

$$\text{“if } \mathbf{x}_1 + \mathbf{x}_2 = (\mathbf{M}_{1,S_0} + \mathbf{M}_{2,S_0})\mathbf{w} \text{ then } \mathbf{y} = \mathbf{N}_{S_1}\mathbf{w}\text{”}.$$

As in the bilateral case, we want to avoid sending the whole vectors

$$[\mathbf{x}_1]_1 = [\mathbf{M}_1]_1\mathbf{w}, \qquad [\mathbf{x}_2]_2 = [\mathbf{M}_2]_2\mathbf{w}, \qquad [\mathbf{y}]_1 = [\mathbf{N}]_1\mathbf{w}$$

This time, we will use a split SSB commitment scheme for the first two and a simple SSB for the last one. Let $([\mathbf{Q}_1]_1, [\mathbf{Q}_2]_2)$ be a key for the former extractable at S_0 and $[\mathbf{F}]_1$ a key for the latter extractable at S_1. Our ultimate goal is to receive (succinct) commitments

$$[\mathbf{c}_1]_1 = [\mathbf{Q}_1(\mathbf{M}_1 + \mathbf{M}_2)]_1\mathbf{w}, \qquad [\mathbf{c}_2]_2 = [\mathbf{Q}_2(\mathbf{M}_1 + \mathbf{M}_2)]_2\mathbf{w}, \qquad [\mathbf{d}_2]_1 = [\mathbf{FN}]_1\mathbf{w}$$

extract them at S_0, S_1, S_1 respectively, and make sure the above knowledge transfer notion is satisfied for the extracted values and a claim $\mathbf{w}$. Similarly to the QASum case, we do not consider simple constraint subsets as in the construction QALin. Rather, our constraints can be described as the pair of sets $S_0, S_1 \subseteq [d]$ each of size at most K_0, K_1 respectively. We extract the first commitment in the position defined by S_1, and the other two by the positions defined by S_2. We next formalize the above discussion and present the construction.

Figure 4.7 Quasi argument QASum for knowledge transfer of sum membership in linear space.

$\mathcal{D}_{\mathsf{par}}(\mathsf{gk}, d, \mathbf{K} = (K_0, K_1), \mathbf{S} = (S_0, S_1))$**:**

$([\mathbf{M}_1]_1, [\mathbf{M}_2]_2, \mathbf{M}_1, \mathbf{M}_2) \leftarrow (\mathcal{M}_1, \mathcal{M}_2)$, $([\mathbf{N}]_1, \mathbf{N}) \leftarrow \mathcal{N}$,
$([\mathbf{Q}_1]_1, [\mathbf{Q}_2]_2, \mathbf{Q}_1, \mathbf{Q}_2, \mathbf{T}_\mathbf{Q}) \leftarrow \mathsf{CS}'.\mathsf{KeyGen}(\mathsf{gk}, n, d, K_0, S_0)$
$([\mathbf{F}]_1, \mathbf{F}, \mathbf{T}_\mathbf{F}) \leftarrow \mathsf{CS}.\mathsf{KeyGen}(\mathsf{gk}_1, n, d, K_1, S_1)$
Output (ρ, θ) where

$$\rho = (\mathsf{gk}, [\mathbf{Q}_1]_2, [\mathbf{Q}_2]_1, [\mathbf{F}]_2, [\mathbf{M}_1]_1, [\mathbf{M}_2]_2, [\mathbf{N}]_1), \qquad \theta = (\mathbf{Q}_1, \mathbf{Q}_2, \mathbf{F}, \mathbf{T}_\mathbf{Q}, \mathbf{T}_\mathbf{F}, \mathbf{M}_1, \mathbf{M}_2, \mathbf{N})$$

$\mathsf{K}(\rho, \theta)$**:**

Set $\mathbf{Q} = \mathbf{Q}_1 + \mathbf{Q}_2$ and sample $\mathbf{K}_0 \leftarrow \mathbb{F}^{\overline{K}_0 \times k}$, $\mathbf{K}_1 \leftarrow \mathbb{F}^{\overline{K}_1 \times k}$, $\mathbf{Z} \leftarrow \mathbb{F}^{n \times k}$, $\mathbf{A} \leftarrow \mathcal{D}_k$ and redefine $\mathbf{A}$ as its first k columns
Compute $[\mathbf{B}]_1 = [\mathbf{M}_1^\top]_1 \mathbf{Q}^\top \mathbf{K}_0 + [\mathbf{N}^\top]_1 \mathbf{F}^\top \mathbf{K}_1 + [\mathbf{Z}]_1$ and $[\mathbf{D}]_2 = [\mathbf{M}_2^\top]_2 \mathbf{Q}^\top \mathbf{K}_0 - [\mathbf{Z}]_2$
$\mathbf{C}_1 = \begin{pmatrix} \mathbf{K}_0 \\ \mathbf{K}_1 \end{pmatrix} \mathbf{A}$ and $\mathbf{C}_2 = \mathbf{K}_0 \mathbf{A}$
Output (srs, τ) where $\mathsf{srs} = ([\mathbf{B}]_1, [\mathbf{D}]_2, [\mathbf{A}]_{1,2}, [\mathbf{C}_1]_2, [\mathbf{C}_2]_1)$ and $\tau = (\mathbf{T}_\mathbf{Q}, \mathbf{T}_\mathbf{F})$

$\mathsf{Prove}(\mathsf{srs}, [\mathbf{c}_1]_1, [\mathbf{c}_2]_2, [\mathbf{d}]_1, \mathbf{w})$**:**

Sample $\mathbf{z} \leftarrow \mathbb{F}^k$ and output $([\boldsymbol{\pi}]_1, [\boldsymbol{\theta}]_2) \leftarrow (\mathbf{w}^\top [\mathbf{B}]_1 - [\mathbf{z}^\top]_1, \mathbf{w}^\top [\mathbf{D}]_2 + [\mathbf{z}^\top]_2)$

$\mathsf{Verify}(\mathsf{srs}, [\mathbf{c}_1]_1, [\mathbf{c}_2]_2, [\mathbf{d}]_1, [\boldsymbol{\pi}]_1, [\boldsymbol{\theta}]_2)$**:**

Output 1 iff $e([\boldsymbol{\pi}]_1, [\mathbf{A}]_2) + e([\boldsymbol{\theta}]_2, [\mathbf{A}]_1) = e\left([\mathbf{c}_1^\top \mid \mathbf{d}^\top]_1, [\mathbf{C}_1]_2\right) + e\left([\mathbf{c}_2^\top]_2, [\mathbf{C}_2]_1\right)$

$\mathsf{Extract}(\tau, [\mathbf{c}_1]_1, [\mathbf{c}_2]_2, [\mathbf{d}]_1, [\boldsymbol{\pi}]_1, [\boldsymbol{\theta}]_2)$**:**

Parse τ as $(\mathbf{T}_\mathbf{Q}, \mathbf{T}_\mathbf{F})$ and output $[\mathbf{x}_1]_1 = \mathbf{T}_\mathbf{Q}^\top [\mathbf{c}_1]_1, [\mathbf{x}_2]_2 = \mathbf{T}_\mathbf{Q}^\top [\mathbf{c}_2]_1, [\mathbf{y}]_1 = \mathbf{T}_\mathbf{F}^\top [\mathbf{d}]_1$

Let $(\mathcal{M}_1, \mathcal{M}_2)$ be some (possibly correlated) witness samplable distributions outputting matrices in $\mathbb{G}_1^{d \times n} \times \mathbb{G}_2^{d \times n}$ and $\mathcal{N}$ be witness samplable distributions outputting matrices in $\mathbb{G}_1^{d \times n}$ for $n, d \in \mathbb{N}$. Let $\mathbf{K} = (K_0, K_1)$ with $K_i \leq d$ and $\mathbf{S} = (S_0, S_1)$ with $|S_i| \subseteq [d]$. Also, let CS be an algebraic SSB commitment scheme and CS′ be a split algebraic commitment key with commitment space $\mathbb{G}_1^{\overline{K}_1}$, $\mathbb{G}_1^{\overline{K}_0} \times \mathbb{G}_2^{\overline{K}_0}$ respectively. The parameter language is

$$\begin{aligned} \mathcal{L}_{\mathsf{par}} = \big\{ & [\mathbf{M}_1]_1, [\mathbf{M}_2]_2, [\mathbf{N}]_1, [\mathbf{Q}_1]_1, [\mathbf{Q}_2]_1, [\mathbf{F}]_2 \mid \exists \mathbf{M}_1, \mathbf{M}_2, \mathbf{N}_1, \mathbf{Q}_1, \mathbf{Q}_2, \mathbf{F} \text{ s.t.} \\ & ([\mathbf{M}_1]_1, [\mathbf{M}_2]_2, \mathbf{M}_1, \mathbf{M}_2) \in \mathsf{Sup}(\mathcal{N}_1, \mathcal{N}_2), ([\mathbf{N}]_1, \mathbf{N}) \in \mathsf{Sup}(\mathcal{N}), \\ & ([\mathbf{Q}_1]_1, [\mathbf{Q}_2]_2, \mathbf{Q}_1, \mathbf{Q}_2, \mathbf{T}_\mathbf{Q}) \in \mathsf{Sup}(\mathsf{CS}'.\mathsf{KeyGen}(\mathsf{gk}, n, K_0, S_1)), \\ & ([\mathbf{F}]_1, \mathbf{F}, \mathbf{T}_\mathbf{F}) \in \mathsf{Sup}(\mathsf{CS}.\mathsf{KeyGen}(\mathsf{gk}_1, n, K_1, S_2)) \big\} \end{aligned}$$

We assume w.l.o.g. that the corresponding relation is efficiently verifiable. The parameters

$\rho = ([\mathbf{M}]_1, [\mathbf{N}_1]_1, [\mathbf{N}_2]_2, [\mathbf{Q}_1]_1, [\mathbf{Q}_2]_1, [\mathbf{F}]_2)$ define the following relations

$$\text{KTSum-}\mathcal{R}^{\text{yes}}_{\rho} = \left\{ [\mathbf{c}_1]_1, [\mathbf{c}_2]_2, [\mathbf{d}]_2, \mathbf{w} \;\middle|\; \begin{pmatrix} \mathbf{c}_1 + \mathbf{c}_2 \\ \mathbf{d} \end{pmatrix} = \begin{pmatrix} (\mathbf{Q}_1 + \mathbf{Q}_2)(\mathbf{M}_1 + \mathbf{M}_2) \\ \mathbf{FN} \end{pmatrix} \mathbf{w} \right\},$$

$$\text{KTSum-}\mathcal{R}^{\text{no}}_{\rho,\mathbf{S}} = \left\{ \begin{matrix} ([\mathbf{c}_1]_1, [\mathbf{c}_2]_2, [\mathbf{d}]_1) \\ ([\mathbf{x}_1]_1, [\mathbf{x}_2]_2, [\mathbf{y}]_1) \\ \mathbf{w} \end{matrix} \;\middle|\; \begin{matrix} \mathbf{x}_1 + \mathbf{x}_2, \mathbf{y} \text{ are valid } S_0, S_1 \text{ openings of} \\ \mathbf{c}_1 + \mathbf{c}_2,\ \mathbf{d}_2 \text{ w.r.t. } \mathbf{Q}_1 + \mathbf{Q}_2, \mathbf{F} \text{ respectively and} \\ \mathbf{x}_1 + \mathbf{x}_2 = (\mathbf{M}_{1,S_0} + \mathbf{M}_{2,S_0})\mathbf{w} \text{ but } \mathbf{y} \neq \mathbf{N}_{S_2}\mathbf{w} \end{matrix} \right\}$$

That is, the partial witness for $\mathbf{S}$ is some valid local openings $[\mathbf{x}_1]_1, [\mathbf{x}_2]_2, [\mathbf{y}]_1$ w.r.t. to $\mathbf{G}, \mathbf{H}, \mathbf{F}$ respectively that satisfy the following: if $\mathbf{x}_1 + \mathbf{x}_2 = (\mathbf{M}_{1,S_1} + \mathbf{M}_{2,S_1})\mathbf{w}$ then it should be the case that $\mathbf{y} = \mathbf{N}_{S_2}\mathbf{w}$ where $\mathbf{w}$ is the promise of the adversary. Note that if S_1 is the empty set the latter relations trivially hold. We present the protocol in Fig 4.7.

We present a detailed security analysis of QASum in Sec. 4.7.2. We next summarize the properties of a specific instantiation of QASum we will use in our delegation construction.

Theorem 20. *Construction* QASum *instantiated with*

(1) target keys of KMPed *of Fig 4.4 for* CS′

(2) MPed *of Fig 4.3 for* CS

(3) $\mathbf{M}_1, \mathbf{M}_2$ *sampled from a distribution that satisfies the* $(\mathcal{U} \otimes \mathcal{V})^\top$-KMDDH *assumption*

(4) $\mathcal{D}_k$ *that satisfies the* $\mathcal{D}_k$-SKerMDH *assumption*

is a quasi argument. Furthermore,

- *Local knowledge soundness holds even against adversaries that know the discrete logarithms of the commitment keys and* $\mathbf{N}$.
- *No-signaling holds even against adversaries that know the discrete logarithms of* $\mathbf{M}_1, \mathbf{M}_2, \mathbf{N}$.

4.4.2.2 Quasi Arguments for Hadamard Products

The main result of [GHR15] was implicitly a quasi-argument for the set of equations $b_i(b_i - 1) = 0$, for all $i \in [d]$. We extend their results to equations of the form $x_i y_i = z_i$, that is $\mathbf{x} \circ \mathbf{y} = \mathbf{z}$ where $\circ$ denotes the Hadamard product.

Consider three $d \times n$ matrices $[\mathbf{U}]_1$, $[\mathbf{V}]_2$, $[\mathbf{W}]_1$, two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}^n$ and vectors $[\mathbf{x}_1]_1$, $[\mathbf{x}_2]_1$, $[\mathbf{y}]_1$. Now, if $S \subseteq [d]$, we want to consider local constraints of the form:

$$\text{“if } [\mathbf{x}_1]_1 = [\mathbf{U}_S]_1\mathbf{a} \text{ and } [\mathbf{x}_2]_2 = [\mathbf{V}_S]_2\mathbf{b} \text{ then } [\mathbf{y}]_1 = [\mathbf{W}_S]_1\mathbf{a} \circ \mathbf{b} \text{ ”.}$$

Figure 4.8 Quasi argument QAHad for knowledge transfer of Hadammard product. Here $\mathbf{D} \in \mathbb{F}^{n \times n^2}$ is the matrix such that $\mathbf{D}(\mathbf{a} \otimes \mathbf{b}) = \mathbf{a} \circ \mathbf{b}$

$\mathcal{D}_{\mathsf{par}}(\mathsf{gk}, d, K, S)$**:**

$([\mathbf{U}]_1, \mathbf{U}) \leftarrow \mathcal{U}$, $([\mathbf{V}]_2, \mathbf{V}) \leftarrow \mathcal{V}$, $([\mathbf{W}]_1, \mathbf{W}) \leftarrow \mathcal{W}$

$([\mathbf{G}]_1, \mathbf{G}, \mathbf{T_G}) \leftarrow \mathsf{CS.KeyGen}(\mathsf{gk}_1, n, d, K, S)$

$([\mathbf{H}]_2, \mathbf{H}, \mathbf{T_H}) \leftarrow \mathsf{CS.KeyGen}(\mathsf{gk}_2, n, d, K, S)$

$([\mathbf{F}]_1, \mathbf{F}, \mathbf{T_F}) \leftarrow \mathsf{CS.KeyGen}(\mathsf{gk}_1, n, d, K, S)$

Output (ρ, θ) where

$$\rho = (\mathsf{gk}, [\mathbf{G}]_1, [\mathbf{H}]_2, [\mathbf{F}]_1, [\mathbf{U}]_1, [\mathbf{V}]_2, [\mathbf{W}]_1), \qquad \theta = (\mathbf{G}, \mathbf{H}, \mathbf{F}, \mathbf{T_G}, \mathbf{T_H}, \mathbf{T_F}, \mathbf{U}, \mathbf{V}, \mathbf{W})$$

$\mathsf{K}(\rho, \theta)$**:**

$(\mathsf{ck}, \mathsf{sk}) \leftarrow \mathsf{KMPed}_t.\mathsf{KeyGen}(\mathsf{gk}, [\mathbf{G}]_1, [\mathbf{H}]_2, \mathbf{G}, \mathbf{H})$ and parse

$$\mathsf{ck} = ([\mathbf{Q}_1]_1, [\mathbf{Q}_2]_2, \mathsf{aux}), \qquad \mathsf{sk} = (\mathbf{Q}_1, \mathbf{Q}_2, \mathbf{T_Q})$$

Sample $\mathbf{R} \in \mathbb{F}^{d^2 \times n^2}$ and set $\mathbf{M}_1 = \mathbf{U} \otimes \mathbf{V} - \mathbf{R}$ and $\mathbf{M}_2 = \mathbf{R}$. Set $\mathbf{N} = \mathbf{WD}$. Also, set

$$\rho_{\mathsf{Sum}} = (\mathsf{gk}, [\mathbf{Q}_1]_1, [\mathbf{Q}_2]_1, [\mathbf{F}]_2, [\mathbf{M}_1]_1, [\mathbf{M}_2]_2, [\mathbf{N}]_1)$$
$$\theta_{\mathsf{Sum}} = (\mathbf{Q}_1, \mathbf{Q}_2, \mathbf{F}, \mathbf{T_Q}, \mathbf{T_F}, \mathbf{M}_1, \mathbf{M}_2, \mathbf{N})$$

Set $(\mathsf{srs}_{\mathsf{Sum}}, \tau_{\mathsf{Sum}}) \leftarrow \mathsf{QASum}(\rho_{\mathsf{Sum}}, \theta_{\mathsf{Sum}})$.

Sample $\mathbf{R}' \leftarrow \mathbb{F}^{\overline{K}^2 \times n^2}$ and set

$$[\mathbf{E}_1]_1 = [\mathbf{Q}_1(\mathbf{U} \otimes \mathbf{V}) - \mathbf{R}']_1, \qquad [\mathbf{E}_2]_2 = [\mathbf{Q}_2(\mathbf{U} \otimes \mathbf{V}) + \mathbf{R}']_2$$

Output $\mathsf{srs} = ([\mathbf{E}_1]_1, [\mathbf{E}_2]_2, \mathsf{srs}_{\mathsf{Sum}})$, $\tau = (\mathbf{T_G}, \mathbf{T_H}, \mathbf{T_F})$

$\mathsf{Prove}(\mathsf{srs}, [\mathbf{x}]_1, [\mathbf{y}]_2, [\mathbf{w}]_1, \mathbf{a}, \mathbf{b})$**:**

Set $[\mathbf{c}_1]_1 = [\mathbf{E}_1]_1(\mathbf{a} \otimes \mathbf{b})$, $[\mathbf{c}_2]_2 = [\mathbf{E}_2]_2(\mathbf{a} \otimes \mathbf{b})$, $[\mathbf{d}]_1 = [\mathbf{w}]_1$

$\pi_{\mathsf{Sum}} = \mathsf{QASum.Prove}(\mathsf{srs}_{\mathsf{Sum}}, [\mathbf{c}_1]_1, [\mathbf{c}_2]_1, [\mathbf{d}]_1, \mathbf{a} \otimes \mathbf{b})$

Output $\pi := ([\mathbf{c}_1]_1, [\mathbf{c}_2]_1, \pi_{\mathsf{Sum}})$

$\mathsf{Verify}(\mathsf{srs}, [\mathbf{u}]_1, [\mathbf{v}]_2, [\mathbf{w}]_1, \pi)$**:**

Compute $[\mathbf{u} \otimes \mathbf{v}]_T$ using the pairing operation and output 1 iff

1. $\mathsf{QASum.Verify}(\mathsf{srs}_{\mathsf{Sum}}, [\mathbf{c}_1]_1, [\mathbf{c}_2]_2, [\mathbf{w}]_1) = 1$ and
2. $[\mathbf{u} \otimes \mathbf{v}]_T = e([\mathbf{c}_1]_1, [1]_2) + e([1]_1, [\mathbf{c}_2]_2)$

$\mathsf{Extract}(\tau, [\mathbf{u}]_1, [\mathbf{v}]_2, [\mathbf{w}]_1, \pi)$**:**

Parse τ as $(\mathbf{T_G}, \mathbf{T_H}, \mathbf{T_F})$ and output $[\mathbf{x}_1]_1 := \mathbf{T_G}^\top[\mathbf{u}]_1$, $[\mathbf{x}_2]_2 := \mathbf{T_H}^\top[\mathbf{v}]_1$, $[\mathbf{y}]_1 := \mathbf{T_F}^\top[\mathbf{w}]_1$.

As in the other quasi arguments, we want to "compress" the claim; that is avoid sending the whole vectors

$$[\mathbf{x}_1]_1 = [\mathbf{U}]_1\mathbf{a}, \qquad [\mathbf{x}_2]_2 = [\mathbf{V}]_2\mathbf{b}[\mathbf{y}]_1 = [\mathbf{W}]_1\mathbf{a} \circ \mathbf{b},$$

Instead, we commit to these elements using three (algebraic) SSB commitments $[\mathbf{G}]_1, [\mathbf{H}]_2, [\mathbf{F}]_1$, which we set extractable at the same set S. Our ultimate goal is to receive (succinct) commitments

$$[\mathbf{c}_1]_1 = [\mathbf{GM}]_1\mathbf{a}, \qquad [\mathbf{c}_2]_1 = [\mathbf{HM}]_2\mathbf{b}, \qquad [\mathbf{d}]_1 = [\mathbf{FM}]_1\mathbf{a} \circ \mathbf{b}$$

extract them at S and make sure the above knowledge transfer notion is satisfied for the extracted values and a claim $\mathbf{a}, \mathbf{b}$. We next formalize this discussion.

Let $\mathcal{U}, \mathcal{V}, \mathcal{W}$ be witness samplable distributions over matrices in $\mathbb{G}_1^{d\times n}, \mathbb{G}_2^{d\times n}$ and $\mathbb{G}_1^{d\times n}$, respectively, for $n, d \in \mathbb{N}$. Let $\mathbf{K} = (K, K)$ with $K \leq d$ and $\mathbf{S} = (S, S)$ with $S \subseteq [d]$ and $S \leq K$. Also let CS be an algebraic SSB commitment scheme with commitment space $\mathbb{G}_\mu^{\overline{K}}$. The parameter language is

$$\begin{aligned}\mathcal{L}_{\mathsf{par}} = \big\{&[\mathbf{U}]_1,[\mathbf{V}]_2,[\mathbf{W}]_1,[\mathbf{G}]_1,[\mathbf{H}]_1,[\mathbf{F}]_2 \mid \exists \mathbf{U},\mathbf{V},\mathbf{W},\mathbf{G},\mathbf{H},\mathbf{F} \text{ s.t.}\\ &([\mathbf{U}]_1,\mathbf{U}) \in \mathsf{Sup}(\mathcal{U}), ([\mathbf{V}]_1,\mathbf{V}) \in \mathsf{Sup}(\mathcal{V}), ([\mathbf{W}]_1,\mathbf{W}) \in \mathsf{Sup}(\mathcal{V})\\ &([\mathbf{G}]_1,[\mathbf{G}]_2,\mathbf{G},\mathbf{T}_\mathbf{G}) \in \mathsf{Sup}(\mathsf{CS.KeyGen}(\mathsf{gk},n,K,S))\\ &([\mathbf{H}]_1,[\mathbf{H}]_2,\mathbf{H},\mathbf{T}_\mathbf{H}) \in \mathsf{Sup}(\mathsf{CS.KeyGen}(\mathsf{gk},n,K,S)),\\ &([\mathbf{F}]_1,\mathbf{F},\mathbf{T}_\mathbf{F}) \in \mathsf{Sup}(\mathsf{CS.KeyGen}(\mathsf{gk},n,K,S))\big\}\end{aligned}$$

We assume w.l.o.g. that the corresponding relation is efficiently verifiable. The parameters $\rho = ([\mathbf{U}]_1, [\mathbf{V}]_2, [\mathbf{W}]_1, [\mathbf{G}]_1, [\mathbf{H}]_1, [\mathbf{F}]_2)$ define the following relations

$$\mathsf{KTHad}\text{-}\mathcal{R}_\rho^{\mathsf{yes}} = \left\{ [\mathbf{u}]_1, [\mathbf{v}]_2, [\mathbf{w}]_1, \mathbf{a}, \mathbf{b} \;\middle|\; \begin{array}{l}\mathbf{u} = \mathbf{GUa}\\ \mathbf{v} = \mathbf{HVb}\\ \mathbf{w} = \mathbf{FW}(\mathbf{a}\circ\mathbf{b})\end{array} \right\},$$

$$\mathsf{KTHad}\text{-}\mathcal{R}_\rho^{\mathsf{no}} = \big\{[\mathbf{U}]_1,[\mathbf{V}]_2,[\mathbf{W}]_1,[\mathbf{G}]_1,[\mathbf{H}]_2,[\mathbf{F}]_1 \left\{ \begin{array}{c}([\mathbf{v}]_1,[\mathbf{u}]_2,[\mathbf{w}]_1)\\ ([\mathbf{x}_1]_1,[\mathbf{x}_2]_2,[\mathbf{y}]_1)\\ \mathbf{a},\mathbf{b}\end{array} \;\middle|\; \begin{array}{l}\mathbf{x}_1,\mathbf{x}_2,\mathbf{y} \text{ are valid } S \text{ openings of}\\ \mathbf{c}_1,\ \mathbf{c}_2,\ \mathbf{d} \text{ w.r.t. } \mathbf{G},\mathbf{H},\mathbf{F} \text{ respectively and}\\ \mathbf{x}_1 = \mathbf{U}_S\mathbf{a}, \mathbf{x}_2 = \mathbf{V}_S\mathbf{b}, \text{ but } \mathbf{y} \neq \mathbf{W}_S(\mathbf{a}\circ\mathbf{b})\end{array} \right\}$$

That is, the partial witness for S is some valid local openings $[\mathbf{x}_1]_1, [\mathbf{x}_2]_2, [\mathbf{y}]_1$ w.r.t. to $\mathbf{G}, \mathbf{H}, \mathbf{F}$ respectively that satisfy the following: if $\mathbf{x}_1 = \mathbf{U}_S\mathbf{a}$ and $\mathbf{x}_2 = \mathbf{V}_S\mathbf{b}$ and then it should be the case that $\mathbf{y} = \mathbf{W}_S\mathbf{c}$ where $\mathbf{c} = \mathbf{a} \circ \mathbf{b}$. Here $\mathbf{a}, \mathbf{b}$ is the promise of the adversary.

We present the protocol in Fig 4.8. Essentially, we first have the prover commit to the Kronecker product $\mathbf{a} \otimes \mathbf{b}$ using a commitment scheme defined by the $\otimes$ operation of CS to itself, and then show that if the split opening of this commitment is $\mathbf{w} = \mathbf{a} \otimes \mathbf{b}$, then the opening of $\mathbf{d}$ is $\mathbf{Dw}$ where $\mathbf{D}$ is the linear operation that outputs $\mathbf{a} \circ \mathbf{b}$ on input $\mathbf{a} \otimes \mathbf{b}$. The former "promise", regarding the Kronecker product, is verified by the pairing operation, while for the latter, QASum is used.

We present a detailed security analysis of QAHad in Sec. 4.7.3. We next summarize the properties of a specific instantiation of QAHad we will use in our delegation construction.

Theorem 21. *Construction* QAHad *instantiated with*

(2) MPed *of Fig 4.3 for* CS

(3) $\mathcal{U}, \mathcal{V}$ *that satisfy the* $(\mathcal{U} \otimes \mathcal{V})^\top$-KMDDH *assumption*

(4) $\mathcal{D}_k$ *that satisfies the* $\mathcal{D}_k$-SKerMDH *assumption*

is a quasi argument. Furthermore,

- *Local knowledge soundness holds even against adversaries that know the discrete logarithms of* $\mathbf{G}, \mathbf{H}, \mathbf{F}, \mathbf{W}$.
- *No-signaling holds even against adversaries that know the discrete logarithms of* $\mathbf{U}, \mathbf{V}, \mathbf{W}$.

4.5 Delegation Construction

We closely follow the blueprint of [GR19]. Consider an arithmetic circuit $C : \mathbb{F}^{n_0} \to \mathbb{F}^{n_d}$. The circuit can be naturally sliced into $d+1$ levels, where level 0 contains the input and level i is formed by a set of n_i multiplication gates, the inputs of which depends on a linear transformation of outputs of previous levels.[16] Let $N_i = \sum_{j=0}^{i} n_i$ and $N = N_d$. Denote by $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i \in \mathbb{F}^{N_i}$ the left, right and output wires of level $1, \ldots, i$ respectively. That is

$$\mathbf{a}_i = \begin{pmatrix} \mathbf{a}_{i-1} \\ \mathbf{D}_i \mathbf{c}_{i-1} \end{pmatrix}, \qquad \mathbf{b}_i = \begin{pmatrix} \mathbf{b}_{i-1} \\ \mathbf{E}_i \mathbf{c}_{i-1} \end{pmatrix}$$

where $\mathbf{D}_i, \mathbf{E}_i \in \mathbb{F}^{n_i \times N_{i-1}}$ are defined by the circuit's linear gates, $\mathbf{a}_0, \mathbf{b}_0$ are of size 0 and $\mathbf{c}_0 = \mathbf{x}$ is the input. Let $\mathbf{D} \in \mathbb{F}^{N - n_0 \times N}$ (resp. $\mathbf{E}$) be the matrix such that the i-th row of $\mathbf{D}$ is $(\mathbf{D}_i | \mathbf{0}_{n_i \times N - N_{i-1}})$. Note that matrices $\mathbf{D}, \mathbf{E}$ are lower triangular. For the multiplication gates, we have the constraints $\mathbf{c}_i = \mathbf{a}_i \circ \mathbf{b}_i$.

Finally, denote $\mathbf{a} = \mathbf{a}_d$, $\mathbf{b} = \mathbf{b}_d$ and $\mathbf{c} = \mathbf{c}_{d-1}$. The evaluation of the circuit is correct if

$$\begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix} = \begin{pmatrix} \mathbf{D} \\ \mathbf{E} \end{pmatrix} \mathbf{c}, \qquad \mathbf{c} = \mathbf{a} \circ \mathbf{b}$$

Next, consider Pedersen commitment keys $\mathbf{U}_i^* \leftarrow \mathbb{F}^{1 \times n_i}$, $\mathbf{V}_i^* \leftarrow \mathbb{F}^{1 \times n_i}$ and $\mathbf{W}_i^* \leftarrow \mathbb{F}^{1 \times n_i}$ and define

$$\mathbf{U}_i = (\mathbf{U}_1^*, \ldots, \mathbf{U}_i^*), \ \mathbf{V}_i = (\mathbf{V}_1^*, \ldots, \mathbf{V}_i^*), \text{ for } i \in [d]$$

[16] We consider w.l.o.g. only linear transformations since we can handle affine ones by including a wire with the value 1 in the input.

$$\mathbf{W}_i = (\mathbf{W}_1^*, \ldots, \mathbf{W}_i^*), \text{ for } i \in [d-1].$$

Consider commitments (represented in $\mathbb{F}$) to left, right and output wires as

$$O_i = \mathbf{W}_i \mathbf{c}_i, \mathbf{O} = \mathbf{W}\mathbf{c}, \qquad L_i = \mathbf{U}_i \mathbf{a}_i = \mathbf{U}\mathbf{a}, \qquad R_i = \mathbf{V}_i \mathbf{b}_i, \mathbf{R} = \mathbf{V}\mathbf{b}$$

where

$$\mathbf{U} = \begin{pmatrix} \mathbf{U}_1^* & & \mathbf{0} \\ \vdots & \ddots & \\ \mathbf{U}_1^* & \cdots & \mathbf{U}_d^* \end{pmatrix}, \qquad \mathbf{V} = \begin{pmatrix} \mathbf{V}_1^* & & \mathbf{0} \\ \vdots & \ddots & \\ \mathbf{V}_1^* & \cdots & \mathbf{V}_d^* \end{pmatrix}, \qquad \mathbf{W} = \begin{pmatrix} \mathbf{W}_1^* & & \mathbf{0} \\ \vdots & \ddots & \\ \mathbf{W}_1^* & \cdots & \mathbf{W}_{d-1}^* \end{pmatrix} \tag{4.5}$$

and $\mathbf{O} = (O_1, \ldots, O_{d-1})^\top, \mathbf{L} = (L_1, \ldots, L_d)^\top, \mathbf{R} = (R_1, \ldots, R_d)^\top$.

We additionally pick $\mathbf{G}, \mathbf{H}, \mathbf{F}$ for computing SSB commitments to vectors of size d and publish $[\mathbf{GU}]_1$, $[\mathbf{HV}]_2$, $[\mathbf{FW}]_2$. The prover gives "commitments" of commitments

$$[\hat{\mathbf{L}}]_1 = [\mathbf{GU}]_1 \mathbf{a}, \qquad [\hat{\mathbf{R}}]_2 = [\mathbf{HV}]_2 \mathbf{b}, \qquad [\hat{\mathbf{O}}]_1 = [\mathbf{FW}]_1 \mathbf{c}$$

and gives a quasi-argument of linear knowledge transfer from $\mathbf{x}, [\mathbf{O}]_1, \mathbf{y}$ to $[\mathbf{L}]_1, [\mathbf{R}]_2$ with the following structure:

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{O} \\ \mathbf{y} \\ \mathbf{L} \\ \mathbf{R} \end{pmatrix} = \begin{pmatrix} \overbrace{\mathbf{I}_{n_0}}^{\text{input}} & \overbrace{\mathbf{0}}^{\text{mid-wires}} & \overbrace{\mathbf{0}}^{\text{output}} \\ \mathbf{0} & \boxed{\mathbf{W}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{n_d} \\ \multicolumn{2}{c}{\boxed{\mathbf{UD}}} & \mathbf{0} \\ \multicolumn{2}{c}{\boxed{\mathbf{VE}}} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{c} \\ \mathbf{y} \end{pmatrix}. \tag{4.6}$$

That is, we can extract $[L_i]_1, [R_i]_2, [O_{i-1}]_1$ and, if we are additionally given $\mathbf{c}_{i-1}$ such that $O_{i-1} = \mathbf{W}_{i-1}\mathbf{c}_{i-1}$, then $L_i = \mathbf{U}_i \mathbf{D}_i \mathbf{c}_i, R_i = \mathbf{V}_i \mathbf{E}_i \mathbf{c}_i$. We also use a quasi-argument of knowledge transfer of the Hadamard product from $[\mathbf{L}]_1, [\mathbf{R}]_2$ to $[\mathbf{O}]_1$. In this case we extract $[L_i]_1, [R_i]_2, [O_i]_1$ and, if we are additionally given $\mathbf{a}_i, \mathbf{b}_i$ such that $L_i = \mathbf{U}_i \mathbf{a}_i$ and $R_i = \mathbf{V}_i \mathbf{b}_i$, then $O_i = \mathbf{W}_i(\mathbf{a}_i \circ \mathbf{b}_i)$.

We need to make one last change that will allow us to take into account the input $\mathbf{x}$ and the claimed output $\mathbf{y}$. Essentially, we make the first and last commitment key (trivially) perfectly binding by using as a commitment key the identity matrix. The security properties still hold in a trivial way (the $\mathbf{I}_{n_0}$-MDDH assumption is perfectly secure). We change accordingly the SSB commitment key, that is we set

$$\mathbf{F}' = \begin{pmatrix} \mathbf{I}_{n_0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{F} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{n_d} \end{pmatrix}$$

Note that the extraction trapdoor remains the same, but the extractor can trivially extract the values corresponding to $\mathbf{x}, \mathbf{y}$ regardless of $\mathbf{F}'$ distribution:. In other words, our commitment

Figure 4.9 Delegation scheme for arithmetic circuits.

$\mathsf{Setup}(1^\kappa, C)$**:**

$\mathsf{gk} \leftarrow \mathcal{G}(1^\kappa)$

From the linear gates of C compute matrices $\mathbf{D}, \mathbf{E}$.

$([\mathbf{F}]_1, \mathbf{F}, \mathbf{T_F}) \leftarrow \mathsf{MPed.KeyGen}(\mathsf{gk}, d-1, 1, \emptyset)$,

$([\mathbf{G}]_1, \mathbf{G}, \mathbf{T_G}) \leftarrow \mathsf{MPed.KeyGen}(\mathsf{gk}, d, 1, \emptyset)$,

$([\mathbf{H}]_1, \mathbf{H}, \mathbf{T_H}) \leftarrow \mathsf{MPed.KeyGen}(\mathsf{gk}, d, 1, \emptyset)$;

Sample $\mathbf{U}, \mathbf{V}, \mathbf{W}$ as in equation 4.5.

Define $\mathbf{W}'$ as the matrix $\mathbf{W}$ augmented with $(\mathbf{I}_{n_0} \mid \mathbf{0})$ and $(\mathbf{0} \mid \mathbf{I}_{n_d})$ as its first and last row.

Let

$$\rho_l = (\mathsf{gk}, [\mathbf{F}']_1, [\mathbf{G}]_1, [\mathbf{H}]_2, [\mathbf{W}']_1, [\mathbf{UD}]_1, [\mathbf{VE}]_2)$$

$$\theta_l = (\mathbf{F}, \mathbf{G}, \mathbf{H}, \mathbf{T_F}, \mathbf{T_G}, \mathbf{T_H}, \mathbf{U}', \mathbf{UD}, \mathbf{VE})$$

where $\mathbf{F}'$ contains rows $(\mathbf{I}_n \mid \mathbf{0} \mid \mathbf{0})$, $(\mathbf{0} \mid \mathbf{F} \mid \mathbf{0})$, $(\mathbf{0} \mid \mathbf{0} \mid \mathbf{I}_{n_d})$.

Let

$$\rho_h = (\mathsf{gk}, [\mathbf{G}]_1, [\mathbf{H}]_2, [\mathbf{F}'']_1, [\mathbf{U}]_1, [\mathbf{V}]_2, [\mathbf{U}]_1)$$

$$\theta_h = (\mathbf{G}, \mathbf{H}, \mathbf{F}, \mathbf{T_G}, \mathbf{T_H}, \mathbf{T_F}, \mathbf{U}, \mathbf{V}, \mathbf{W})$$

where $\mathbf{F}''$ contains the rows $(\mathbf{F} \mid \mathbf{0})$, $(\mathbf{0} \mid \mathbf{I}_{n_d})$.

Sample $\mathsf{srs}_l \leftarrow \mathsf{QABLin.K}(\rho_l, \theta_l)$, $\mathsf{srs}_h \leftarrow \mathsf{QAHad.K}(\rho_h, \theta_h)$

Output $\mathsf{srs} := ([\mathbf{GU}]_1, [\mathbf{HV}]_2, [\mathbf{FW}]_1, \mathsf{srs}_{\mathsf{QALin}}, \mathsf{srs}_h)$

$\mathsf{Prove}(\mathsf{srs}, \mathbf{x}, \mathbf{y})$**:**

Evaluate $C(\mathbf{x})$ to obtain values for the wires $\mathbf{a}, \mathbf{b}, \mathbf{c}$.

Compute $[\hat{\mathbf{L}}]_1 = [\mathbf{GU}]_1\mathbf{a}, [\hat{\mathbf{R}}]_2 = [\mathbf{HV}]_2\mathbf{b}, [\hat{\mathbf{O}}]_1 = [\mathbf{FW}]_1\mathbf{c}$.

$$\pi_l \leftarrow \mathsf{QABLin.Prove}\left(\mathsf{srs}_l, \begin{pmatrix}\mathbf{x}\\ [\hat{\mathbf{O}}]_1 \\ \mathbf{y}\end{pmatrix}, [\hat{\mathbf{L}}]_1, [\hat{\mathbf{R}}]_2, (\mathbf{x}, \mathbf{c}, \mathbf{y})\right).$$

$$\pi_h \leftarrow \mathsf{QAHad.Prove}\left(\mathsf{srs}_h, [\hat{\mathbf{L}}]_1, [\hat{\mathbf{R}}]_2, \begin{pmatrix}[\hat{\mathbf{O}}]_1 \\ \mathbf{y}\end{pmatrix}, \mathbf{a}, \mathbf{b}\right).$$

Return $\pi = \left([\hat{\mathbf{O}}]_1, [\hat{\mathbf{L}}]_1, [\hat{\mathbf{R}}]_2, \pi_l, \pi_h\right)$.

$\mathsf{Verify}(\mathsf{srs}, (\mathbf{x}, \mathbf{y}), \pi)$**:**

Output 1 iff the following tests succeed

$$\mathsf{QABLin.Verify}(\mathsf{srs}_l, \begin{pmatrix}\mathbf{x}\\ [\hat{\mathbf{O}}]_1 \\ \mathbf{y}\end{pmatrix}, [\hat{\mathbf{L}}]_1, [\hat{\mathbf{R}}]_2, \pi_l) = 1$$

$$\mathsf{QAHad.Verify}(\mathsf{srs}_h, [\hat{\mathbf{L}}]_1, [\hat{\mathbf{R}}]_2, \begin{pmatrix}[\hat{\mathbf{O}}]_1 \\ \mathbf{y}\end{pmatrix}, \pi_h) = 1$$

key is always perfectly binding in the first n_0 and n_d coordinates. We denote with $\mathbf{W}'$ the modified matrix where we change the first and last rows with $(\mathbf{I}_{n_0} \mid \mathbf{0})$ and $(\mathbf{0} \mid \mathbf{I}_{n_d})$ respectively. Therefore, if $\mathbf{O} = \mathbf{W}'\mathbf{c}$, we get that $O_0 = \mathbf{x}$ and $O_d = \mathbf{y}$.

We present the construction in Fig 4.9.

Theorem 22. *When instantiating* QABLin, QAHad *as described in Thm. 19 and Thm. 21 respectively, construction of Fig. 4.9 is a delegation scheme.*

Proof. Let Game_0 be the soundness game:

Game_0**:** This is the soundness game. The output of Game_0 is 1 iff on input $\mathsf{srs} \leftarrow \mathsf{Setup}(1^\kappa, C)$, the adversary outputs $\mathbf{x}, \mathbf{y}, \pi \leftarrow \mathcal{A}(\mathsf{srs})$ such that $C(\mathbf{x}) \neq \mathbf{y}$ and the proof verifies, namely $\mathsf{Verify}(\mathsf{srs}, \mathbf{x}, \mathbf{y}, \pi) = 1$.

In what follows we use the fact that the commitment keys corresponding to $[O_0]_1$ and $[O_d]_1$ are the identity matrices; thus they are trivially extractable. Therefore, the bilateral quasi argument is sound since it satisfies the soundness conditions (MDDH is trivially hard for the identity matrix). This is used in the same way as [GR19].

For $i \in [d]$, $j \in [0, d]$ and S_1, S_2 sets of sizes at most 1, consider the following games, capturing possible bad events:

$\mathsf{BadO}_{j,S_1,S_2}$**:** As Game_0 with the following difference: we sample commitment keys that make srs_h extractable at $\mathbf{S} = (S_1, S_2)$ and a corresponding trapdoor τ. The output of the game is iff either $S_2 \neq \{j\}$ or $[O_j]_1 \neq [\mathbf{W}_j^*]_1 \mathbf{c}_j$, where:

- $\mathbf{c}_j$ is computed by honestly executing $C(\mathbf{x})$ and
- $[O_j]_1$ is extracted from the adversary's proof π_h as

$$[O_j]_1 \leftarrow \mathsf{QAHad.Extract}(\tau, [\hat{\mathbf{L}}]_1, [\hat{\mathbf{R}}]_2, [\hat{\mathbf{O}}]_1, \pi_h)$$

$\mathsf{BadLR}_{i,S_1,S_2}$**:** As Game_0 with the difference: we sample commitment keys that make srs_l extractable at $\mathbf{S} = (S_1, S_2)$ and a corresponding trapdoor τ. The output of the game is 1 iff either $S_1 \neq \{i\}$ or $[L_i]_1 \neq [\mathbf{U}_i^*]_1 \mathbf{a}_i$ or $[R_i]_1 \neq [\mathbf{V}_i^*]_1 \mathbf{b}_i$ where:

- $\mathbf{a}_i, \mathbf{b}_i$ are computed by honestly executing $C(\mathbf{x})$ and
- $[L_i]_1, [R_i]_2$ are extracted from the adversary's proof π_l as

$$([L_i]_1, [R_i]_1) \leftarrow \mathsf{QABLin.Extract}(\tau, [\hat{\mathbf{L}}]_1, [\hat{\mathbf{R}}]_2, [\hat{\mathbf{O}}]_1, \pi_l)$$

Now, let E be the event where the output $(\mathbf{x}, \mathbf{y}, \pi) \leftarrow \mathcal{A}(\mathsf{srs})$ satisfies $\mathsf{Verify}(\mathsf{srs}, \mathbf{x}, \mathbf{y}, \pi) = 1$. We define $O_d = \mathbf{y}$ and $\mathbf{W}_d = (\mathbf{0}_{n_d \times N - n_d} | \mathbf{I}_{n_d})$ so that $\mathsf{Game}_0 = \mathsf{BadO}_{d,\emptyset,\{d\}} \wedge E$. We also define $\mathsf{BadO}_i = \mathsf{BadO}_{i,\emptyset,\{i\}}$.

We next prove the following claim, that states that for each level j, the probability of extracting a commitment to the output wires O_i that is different from an honestly computed one $\mathbf{W}_j^* \mathbf{c}_j$ ($\mathsf{BadO}_i = 1$) is negligible. Note that is enough to conclude the proof since in the last level, this probability corresponds to the soundness game.

Claim 1. Let $\mathcal{A}$ be an adversary against soundness of the delegation scheme. Then, for all $i \in \{1, \ldots, d\}$ $\Pr[\mathsf{BadO}_i = 1 \mid E] = p_i$. We claim that

$$\Pr\left[\mathsf{BadO}_i = 1 \mid E\right] \leq \mathsf{negl}(\kappa)$$

Proof. The proof of the claim is by induction over i. In the inductive case we show that

$$\begin{aligned}
\Pr[\mathsf{BadO}_i^{\mathcal{A}} = 1 \mid E] &\approx \Pr[\mathsf{BadO}^{\mathcal{A}}_{i,(\{i+1\},\{i\})} = 1 \mid E] \\
&\approx \Pr[\mathsf{BadLR}^{\mathcal{A}}_{i,(\{i+1\},\{i\})} = 1 \mid E] \\
&\approx \Pr[\mathsf{BadLR}^{\mathcal{A}}_{i,(\{i+1\},\{i+1\})} = 1 \mid E] \\
&\approx \Pr[\mathsf{BadO}^{\mathcal{A}}_{i+1,(\{i+1\},\{i+1\})} = 1 \mid E] \\
&\approx \Pr[\mathsf{BadO}^{\mathcal{A}}_{i+1} = 1 \mid E]
\end{aligned}$$

where $p_1 \approx p_2$ is defined as $|p_1 - p_2| \leq \mathsf{negl}(\kappa)$. For the base of the induction, note that $\Pr\left[\mathsf{BadO}_0 = 1 \mid E\right] = 0$ since this condition is directly verified and the proof is accepting. Now we show that each $\approx$ is indeed negligible. Note that ρ_h can be computed from ρ_l and vice-versa.

$\mathsf{BadO}_i, \mathsf{BadO}_{i,(\{i+1\},\{i\})}$**:** Consider the sets $\mathbf{S}_1 = (\emptyset, \{i\})$ and $\mathbf{S}_2 = (\{i+1\}, \{i\})$. We show that the output of the games relative to $\mathcal{A}$ are computationally indistinguishable by reducing to the no-signaling property of QABLin.

We construct an adversary $\mathcal{D}$ against no-signaling extraction of QABLin. By Thm. 19, QABLin is no-signaling even when $\mathcal{D}$ is given $\mathbf{U}, \mathbf{V}, \mathbf{W}$. Using these, $\mathcal{D}$ can compute srs_h of QAHad (cf. Lemma 4). It then runs $\mathcal{A}(\mathsf{srs})$ until it outputs $(\mathbf{x}, \mathbf{y}^*, [\hat{\mathbf{O}}]_1, [\hat{\mathbf{L}}]_1, [\hat{\mathbf{R}}]_2, \pi_l, \pi_h)$, and then $\mathcal{D}$ outputs

$$\begin{matrix} \mathbf{x} \\ [\hat{\mathbf{O}}]_1,\ [\hat{\mathbf{L}}]_1,\ [\hat{\mathbf{R}}]_2,\ \pi_l. \\ \mathbf{y}^* \end{matrix}$$

It gets the extracted value for the intersection of the two sets namely $[O_i]_1$. It outputs 1 if and $[O_i]_1 \neq \mathbf{W}_i^* \mathbf{c}_i$) and and otherwise 0. Note that by the induction hypothesis, in the former game this event happens only with negligible probability. Thus, this is a winning strategy against the no-signaling challenge. Thus, it should be the case that

$$\Pr[\mathsf{BadO}_{i,(\{i+1\},\{i\})} = 1 \mid E] \leq \mathsf{negl}(\kappa)$$

$\mathsf{BadO}_{i,(\{i+1\},\{i\})}, \mathsf{BadLR}_{i+1,(\{i+1\},\{i\})}$**:** We build an adversary $\mathcal{B}$ against the local knowledge soundness of QABLin. By Thm. 19, QABLin has local knowledge soundness even when $\mathcal{B}$ is given $\mathbf{G}, \mathbf{H}, \mathbf{F}, \mathbf{U}, \mathbf{V}$. On input srs_l and these values, $\mathcal{B}$ can computes srs_h, (cf. Lemma 4). Then, it runs $\mathcal{A}(\mathsf{srs})$ until it outputs $\mathbf{x}, \mathbf{y}^*, \pi$ and then $\mathcal{B}$ outputs

$$\begin{matrix}\mathbf{x}\\ [\hat{\mathbf{O}}]_1,\ [\mathbf{L}]_1,\ [\mathbf{R}]_2,\ \pi_l\\ \mathbf{y}^*\end{matrix}$$

Now, conditioned on E, if the events $\neg\mathsf{BadO}_{i+1,(\{i+1\},\{i\})}$ and $\mathsf{BadLR}_{i+1,(\{i\},\{i\})}$ happen, it holds that π_l verifies, $[O_i]_1 = \mathbf{W}_i\mathbf{c}_i$ and

$$[L_{i+1}]_1 \neq \mathbf{U}_{i+1}\mathbf{a}_{i+1} \text{ or } [R_{i+1}]_1 \neq \mathbf{V}_{i+1}\mathbf{b}_{i+1}$$

which breaks local extractability of QABLin. This happens only with negligible probability, so

$$\Pr[\Pr[\mathsf{BadLR}_{i,(\{i+1\},\{i\})} = 1 \mid E] \leq \mathsf{negl}(\kappa)$$

$\mathsf{BadLR}_{i+1,(\{i+1\},\{i\})}, \mathsf{BadLR}_{i+1,(\{i+1\},\{i+1\})}$**:** Similarly as the case $\mathsf{BadO}_i, \mathsf{BadO}_{i,(\{i+1\},\{i\})}$, but we need to transition between sets $(\{i+1\},\{i\}) \to (\{i+1\},\emptyset) \to (\{i+1\},\{i+1\})$. We use twice the no-signaling property of QAHad and exploit the fact that we can build srs_l using $\mathbf{U}, \mathbf{V}, \mathbf{W}$ (cf. Lemma 2). Therefore,

$$\Pr[\mathsf{BadLR}_{i,(\{i+1\},\{i+1\})} = 1 \mid E] \leq \mathsf{negl}(\kappa)$$

$\mathsf{BadLR}_{i,(\{i+1\},\{i+1\})}, \mathsf{BadO}_{i+1,(\{i+1\},\{i+1\})}$**:** We build an adversary $\mathcal{B}$ against the knowledge soundness of QAHad. By Thm. 21, local knowledge soundness holds even when $\mathcal{B}$ knows the discrete logarithms $\mathbf{G}, \mathbf{H}, \mathbf{F}, \mathbf{W}$. Using these values, it computes srs_l, as in (cf. Lemma 2). Then, it runs $\mathcal{A}(\mathsf{srs})$ until it outputs $\mathbf{x}, \mathbf{y}^*, \pi$ and then $\mathcal{B}$ outputs

$$[\mathbf{L}]_1, [\mathbf{R}]_2, [\begin{matrix}\hat{\mathbf{O}}\\ \mathbf{y}^*\end{matrix},]_1 [\pi_h]_1$$

Now, conditioned on E, if the events $\neg\mathsf{BadLR}^{\mathcal{A}}_{i+1,(\{i+1\},\{i+1\})}$ and $\mathsf{BadO}^{\mathcal{A}}_{i+1,(\{i+1\},\{i+1\})}$ happen, it holds that π_h verifies and also

$$[L_{i+1}]_1 = \mathbf{U}_{i+1}\mathbf{a}_{i+1} \text{ and } [R_{i+1}]_1 = \mathbf{V}_{i+1}\mathbf{b}_{i+1} \text{ but } [O_{i+1}]_1 \neq \mathbf{W}_{i+1}\mathbf{c}_{i+1}$$

which breaks local extractability of QAHad. Therefore,

$$\Pr[\mathsf{BadO}_{i+1,(\{i+1\},\{i+1\})} = 1 \mid E] \leq \mathsf{negl}(\kappa)$$

$\mathsf{BadO}_{i+1,(\{i+1\},\{i+1\})}, \mathsf{BadO}_{i+1}$**:** Identical to the case $\mathsf{BadO}_i, \mathsf{BadO}_{i,(\{i+1\},\{i\})}$. We can thus bound

$$\Pr[\mathsf{BadO}_{i+1} = 1 \mid E] \leq \mathsf{negl}(\kappa)$$

□

Using the claim, we conclude that

$$\Pr[\mathsf{BadO}_d = 1 \mid E] = \Pr[\mathsf{Game}_0 = 1] \leq \mathsf{negl}(\kappa)$$

□

Efficiency. The size of the srs is $(6N^2 + 6N + 24)\mathbb{G}_1$ elements and $(6N^2 + 4N + 36)\mathbb{G}_2$ elements and computing it is dominated by the same number of group exponentiations in $\mathbb{G}_1, \mathbb{G}_2$ respectively; the prover is dominated by $6N^2+6N$ exponentiations in $\mathbb{G}_1$ and $6N^2+2N$ exponentiations in $\mathbb{G}_2$ and produces a proof of size $12\mathbb{G}_1+10\mathbb{G}_2$ group elements; verifying a proof requires 36 pairing operations. The size of the proof can be reduced to $10\mathbb{G}_1+8\mathbb{G}_2$ if we combine the linear argument with the second linear argument used by the Hadamard quasi argument.

4.6 Applications

In this section we show how to use our delegation scheme to (1) get a NIZK argument for NP in the pre-processing model where the size of the proof is linear in the size of the NP witness and independent of the computation size, in spite of most NIZK constructions under standard assumptions; (2) a zk-SNARK with quantitatively weaker assumptions and (3) compact NIZK for NP with proof size proportional to the witness.

4.6.1 NIZK arguments for NP.

Let CS_E be an algebraic commitment scheme –namely compatible with the Groth-Sahai proof system [GS08] –which is hiding and extractable. We can express the verification algorithm Del.Verify as a set of pairing product equation. The idea to construct a NIZK is the following: let C be an arithmetic circuit that takes public input $\mathbf{x}$ and secret input $\mathbf{w}$, and let $\mathsf{srs}_{\mathsf{Del}}$ be a srs for the delegation of computation of C. The prover commits to $\mathbf{w}$ and the group elements defining the proof of the delegation using the extractable commitment and gives a Groth-Sahai proof that the set of verification equations are satisfied w.r.t. the opening of the commitment. Now, if CS_E is extractable, we can extract the witness $\mathbf{w}$, and if the circuit is not satisfied w.r.t. $\mathbf{x}, \mathbf{w}$ we can break adaptive soundness of delegation scheme Del. We present the scheme in Fig 4.10.

Theorem 23. *Let* CS_E *be an algebraic commitment scheme that is hiding and extractable,* GS *the Groth-Sahai proof system of [GS08] and* Del *the delegation scheme of Fig. 4.9. Then, construction of Fig. 4.10 is a NIZK argument of knowledge. Furthermore, for every adversary* $\mathcal{A}$ *against knowledge soundness there exist adversaries* $\mathcal{B}_1, \mathcal{B}_2$ *against extractability of* CS_E *and against soundness of* Del *respectively such that* $\mathcal{A}(\mathcal{A}) \leq \mathcal{A}^{\mathsf{CS}_E}_{ext}(\mathcal{B}_1) + \mathcal{A}^{\mathsf{Del}}_{\mathsf{Snd}}(\mathcal{B}_2)$.

Figure 4.10 NIZK argument for **NP**. CS_E is an algebraic commitment, GS is the Groth-Sahai proof system of [GS08] and Del the delegation scheme of Fig. 4.9

$\mathsf{Setup}(\mathsf{gk}, C)$**:** Let C an arithmetic circuit which on public input $\mathbf{x}$ size n_x and secret input $\mathbf{w}$ size n_w outputs $\mathbf{y}$ of size n_d.

$\mathsf{ck}_w \leftarrow \mathsf{CS}_E(\mathsf{gk}, n_w)$; $\mathsf{srs}_{\mathsf{Del}} \leftarrow \mathsf{Del.Setup}(\mathsf{gk}, C)$

$\mathsf{srs}_{\mathsf{GS}} \leftarrow \mathsf{GS.Setup}(\mathsf{gk})$

Output $\mathsf{srs} = (\mathsf{ck}_w, \mathsf{srs}_{\mathsf{Del}}, \mathsf{srs}_{\mathsf{GS}})$.

$\mathsf{Prove}(\mathsf{srs}, \mathbf{w}, \mathbf{x}, \mathbf{y})$**:**

Parse $\mathsf{srs} = (\mathsf{ck}_w, \mathsf{srs}_{\mathsf{Del}}, \mathsf{srs}_{\mathsf{GS}})$

Compute $\pi \leftarrow \mathsf{Del}(\mathsf{srs}_{\mathsf{Del}}, (\mathbf{x}, \mathbf{w}), \mathbf{y})$ and $\mathsf{c}_w = \mathsf{CS}_E.\mathsf{Com}(\mathbf{w}; \mathbf{r})$

Denote ϕ_{GS} the system of pairing product equations that contain

1. The equations defined by $\mathsf{Del.Verify}(\mathsf{srs}, (\mathbf{x}, \mathbf{w}), \mathbf{y}, \pi) = 1$, where the unknowns are $\mathbf{w}$ and π
2. The equations defined by $\mathsf{c}_w = \mathsf{CS}_E.\mathsf{Com}(\mathsf{ck}_w, \mathbf{w}; \mathbf{r})$, where the unknowns are $\mathbf{w}$ and $\mathbf{r}$

$\pi_{\mathsf{GS}} \leftarrow \mathsf{GS.Prove}(\mathsf{srs}_{\mathsf{GS}}, \phi_{\mathsf{GS}}, (\mathbf{w}, \mathbf{r}))$

Output $\pi \leftarrow (\mathsf{c}_w, \pi_{\mathsf{GS}})$.

$\mathsf{Verify}(\mathsf{srs}, (\mathbf{x}, \mathbf{y}), \pi)$**:**

Parse $\mathsf{srs} = (\mathsf{ck}_w, \mathsf{srs}_{\mathsf{Del}}, \mathsf{srs}_{\mathsf{GS}})$ and $\pi = (\mathsf{c}_w, \pi_{\mathsf{GS}})$.

Output 1 iff $\mathsf{GS.Verify}(\mathsf{srs}_{\mathsf{GS}}, \phi_{\mathsf{GS}}, \pi_{\mathsf{GS}}) = 1$

Proof. Completeness follows by the correctness of CS_E, and completeness of GS, Del. Computational zero knowledge follows from the computational zero-knowledge of GS and the hiding property of CS_E. For knowledge soundness, we show how we can extract a valid witness given an accepting proof. In what follows, let $\mathcal{E}_{\mathsf{CS}}$ be the extractors for CS_E. The NIZK extractor $\mathcal{E}_{\mathcal{A}}(\mathsf{srs}, \mathbf{x}, \mathbf{y}, \pi = (c_w, \pi_{\mathsf{GS}}))$ simply outputs $(\mathbf{w}, \pi) \leftarrow \mathcal{E}_{\mathsf{CS}}(\mathsf{ck}_w, c_w)$. Now, we claim that this a valid witness except with negligible probability. It is enough to note that if it is not, there are three possible cases:

1. The extractor $\mathcal{E}_{\mathsf{CS}}$ failed which contradicts extractability of CS_E.
2. The extracted solutions $\mathbf{w}, \pi, \mathbf{r}$ are not solutions to ϕ_{GS}, contradicting perfect soundness of GS since the proof verifies.
3. $\mathbf{y} \neq C(\mathbf{x}, \mathbf{w})$. We can extract the solution $\mathbf{w}, \pi, \mathbf{r}$ and it must hold that

$$\mathsf{Del.Verify}(\mathsf{srs}, (\mathbf{x}, \mathbf{w}), \mathbf{y}, \pi) = 1$$

contradicting adaptive soundness of Del.

□

As for efficiency, and specifically proof size, noting that the Groth-Sahai proof gives only a constant, multiplicative overhead to the proof –which is of constant size–, its size is dominated by the size of CS_E. Depending on the choice of CS_E we can get qualitatively different constructions. We consider the following cases:

(i) For a NIZK argument of knowledge under falsifiable assumptions, we can extend our result to apply to boolean circuits instead of arithmetic ones by arithmetizing the different types of gates e.g. as in [DFGK14]. We can then use commitments for boolean vectors that are extractable in the field under falsifiable assumptions such as Groth-Sahai commitments or using the techniques of [GHR15]. The proof size in this case is $O(\kappa\,|\mathbf{w}|)$ where $\mathbf{w}$ is the secret input. Since fully succinct algebraic extractable commitments that allow extraction in the field are unknown to exist under falsifiable assumptions, we cannot achieve a (concretely more efficient) NIZK AoK for arithmetic circuits.

(ii) We can use succinct extractable commitments based on knowledge assumptions, yielding a SNARK of constant proof size. Additionally, since the committed value is the secret input and not the full wire assignment we get a quantitatively smaller assumption size. For example, in case of q-power knowledge of exponent assumption (q-KoE assumption) used in [DFGK14], we use only the n_w-KoE while [DFGK14] requires the larger (and hence stronger) $|C|$-KEA.

(iii) To construct a compact NIZK where the proof size is $O(|\mathbf{w}|) + \mathrm{poly}(\kappa)$ we can follow the ideas of [KNYY19; KNYY20]. We use a symmetric encryption scheme $\mathsf{SE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with additive overhead in the ciphertexts. That is, $|\mathsf{SE.Enc}(\mathsf{sk}, \mathbf{w})| = O(|\mathbf{w}|) + \mathrm{poly}(\kappa)$. We use the NIZK from figure 4.10, instantiated with the commitment scheme from (i), for showing knowledge of some key $K \in \mathrm{Im}(\mathsf{SE.KeyGen})$ such that $C'(K, D) = 1$, where K is the secret input, D the public input, and $C'(K, D) = C(\mathsf{SE.Dec}(K, D))$. To prove that $C(\mathbf{w}) = 1$ the prover picks $K \leftarrow \mathsf{SE.KeyGen}(1^\kappa)$ and computes $D \leftarrow \mathsf{SE.Enc}(K, \mathbf{w})$ together with a proof π that $C'(K, D) = 1$. The verifier on input srs, D and π outputs 1 if π is a valid proof for D. In contrast with [KNYY19; KNYY20], by the nature of the underlying non-compact NIZK scheme we use, we don't require SE.Dec to be in $\mathbf{NC}^1$.

4.7 Deferred Proofs

4.7.1 Security Analysis of QABLin

Theorem 24. *Let $\mathcal{M}, \mathcal{N}_1, \mathcal{N}_2$ be witness samplable distributions, $\mathcal{D}_k$ be a matrix distribution and* CS *an algebraic SSB commitment with perfect completeness. Then,* QABLin *is complete and its local knowledge soundness reduces to knowledge transfer soundness of* KTLin.

Proof. For completeness, we have that

$$\begin{aligned}(\mathbf{c}^\top \mid \mathbf{d}_1^\top)\mathbf{C}_1 + \mathbf{d}_2^\top \mathbf{C}_2 &= (\mathbf{c}^\top \mid \mathbf{d}_1^\top)\begin{pmatrix}\mathbf{K}_0\\ \mathbf{K}_1\end{pmatrix}\mathbf{A} + \mathbf{d}_2^\top \mathbf{K}_2 \mathbf{A}\\ &= (\mathbf{c}^\top \mathbf{K}_0 + \mathbf{d}_1^\top \mathbf{K}_1 + \mathbf{d}_2^\top \mathbf{K}_2)\mathbf{A}\\ &= \left(\mathbf{w}^\top \mathbf{M}^\top \mathbf{G}^\top \mathbf{K}_0 + \mathbf{w}^\top \mathbf{N}_1^\top \mathbf{H}^\top \mathbf{K}_1 + \mathbf{w}^\top \mathbf{N}_2^\top \mathbf{F}^\top \mathbf{K}_2\right)\mathbf{A}\\ &= \left(\mathbf{w}^\top (\mathbf{M}^\top \mathbf{G}^\top \mathbf{K}_0 + \mathbf{N}_1^\top \mathbf{H}^\top \mathbf{K}_1) + \mathbf{w}^\top \mathbf{N}_2^\top \mathbf{F}^\top \mathbf{K}_2\right)\mathbf{A}\\ &= \mathbf{w}^\top \mathbf{B}\mathbf{A} + \mathbf{w}^\top \mathbf{D}\mathbf{A}\\ &= \boldsymbol{\pi}\mathbf{A} + \boldsymbol{\theta}\mathbf{A}\end{aligned}$$

Local knowledge transfer follows using almost an identical argument to Thm. 16 and reducing to knowledge transfer of linear KTA Argument of [GR19] presented in Fig. 4.1. Given an adversary $\mathcal{A}$ breaking local knowledge soundness of QABLin we construct another adversary $\mathcal{B}_S$ that breaks soundness of the argument KW-KTLin for matrices $[\mathbf{M}_{S_1}]_1$, $[\mathbf{N}_{1,S_2}]_1$ and $[\mathbf{N}_{2,S_2}]_2$. $\mathcal{B}_S$ works as follows: it takes input $(\rho^\dagger, \mathsf{srs}^\dagger)$ where

$$\rho^\dagger := (\mathsf{gk}, [\mathbf{M}_{S_1}]_1, [\mathbf{N}_{1,S_2}]_1, [\mathbf{N}_{2,S_2}]_2)$$

$$\mathsf{srs}^\dagger := ([\mathbf{B}^\dagger]_1, [\mathbf{D}^\dagger]_2, [\mathbf{A}]_{1,2}, [\mathbf{C}_1^\dagger]_2, [\mathbf{C}_2^\dagger]_1)$$

and does the following:

- $([\mathbf{G}]_1, \mathbf{G}, \mathbf{T}_\mathbf{G}) \leftarrow \mathsf{CS.KeyGen}(gk_1, d, K_1, S_1)$.
- $([\mathbf{H}]_1, \mathbf{H}, \mathbf{T}_\mathbf{H}) \leftarrow \mathsf{CS.KeyGen}(gk_1, d, K_2, S_2)$.
- $([\mathbf{F}]_2, \mathbf{F}, \mathbf{T}_\mathbf{F}) \leftarrow \mathsf{CS.KeyGen}(gk_2, d, K_2, S_2)$.
- It samples $\mathbf{M}_{\bar{S}_1}, \mathbf{N}_{1,\bar{S}_2}, \mathbf{N}_{2,\bar{S}_2}$, such that $\mathbf{M} = \mathbf{P}_{S_1}\begin{pmatrix}\mathbf{M}_{S_1}\\ \mathbf{M}_{\bar{S}_1}\end{pmatrix}$, $\mathbf{N}_1 = \mathbf{P}_{S_2}\begin{pmatrix}\mathbf{N}_{1,S_2}\\ \mathbf{N}_{1,\bar{S}_2}\end{pmatrix}$, $\mathbf{N}_2 = \mathbf{P}_{S_2}\begin{pmatrix}\mathbf{N}_{2,S_2}\\ \mathbf{N}_{2,\bar{S}_2}\end{pmatrix}$.
- $\mathbf{R}_0 \leftarrow \mathbb{F}^{\bar{K}_1 \times k}$; $\mathbf{R}_1 \leftarrow \mathbb{F}^{\bar{K}_2 \times k}$; $\mathbf{R}_2 \leftarrow \mathbb{F}^{\bar{K}_2 \times k}$.
- It computes $[\mathbf{B}]_1 := [\mathbf{B}^\dagger]_1 + [\mathbf{M}]_1^\top \mathbf{G}^\top \mathbf{R}_0 + [\mathbf{N}_1]_1^\top \mathbf{H}^\top \mathbf{R}_1$ and $[\mathbf{D}]_2 := [\mathbf{D}^\dagger]_2 + [\mathbf{N}_2]_2^\top \mathbf{F}^\top \mathbf{R}_2$
- It computes $[\mathbf{C}_1]_2 := \begin{pmatrix}\mathbf{T}_\mathbf{G} & \mathbf{0}\\ \mathbf{0} & \mathbf{T}_\mathbf{H}\end{pmatrix}[\mathbf{C}_1^\dagger]_2 + \begin{pmatrix}\mathbf{R}_0\\ \mathbf{R}_1\end{pmatrix}[\mathbf{A}]_2$ and $[\mathbf{C}_2]_1 := \mathbf{T}_\mathbf{F}[\mathbf{C}_2^\dagger]_1 + \mathbf{R}_2[\mathbf{A}]_1$.
- It sets

$$\rho := ([\mathbf{G}]_1, [\mathbf{H}]_1, [\mathbf{F}]_2, [\mathbf{M}]_1, [\mathbf{N}_1]_1, [\mathbf{N}_2]_2),$$

$$\mathsf{srs} := ([\mathbf{B}]_1, [\mathbf{D}]_2, [\mathbf{A}]_{1,2}, [\mathbf{C}_1]_2, [\mathbf{C}_2]_1)$$

It then executes $\mathcal{A}(\rho, \mathsf{srs})$ until it outputs a statement $([\mathbf{c}]_1, [\mathbf{d}_1]_1, [\mathbf{d}_2]_2, \mathbf{w})$ together with an accepting proof $[\boldsymbol{\pi}]_1, [\boldsymbol{\theta}]_2$. Given an accepting proof $\mathcal{B}_{\mathbf{S}}$ sets $[\mathbf{x}^\dagger]_1 = \mathbf{T_G}[\mathbf{c}]_1, [\mathbf{y}_1^\dagger]_1 = \mathbf{T_H}[\mathbf{d}_1]_1, [\mathbf{y}_2^\dagger]_2 = \mathbf{T_F}[\mathbf{d}_2]_2$, $[\boldsymbol{\pi}^\dagger]_1 = [\boldsymbol{\pi}]_1 - [\mathbf{c}]_1^\top \mathbf{R}_0 - [\mathbf{d}_1]_1^\top \mathbf{R}_1$ and $[\boldsymbol{\theta}^\dagger]_2 = [\boldsymbol{\theta}]_1 - [\mathbf{d}_2]_2^\top \mathbf{R}_2$. It outputs $\left(([\mathbf{x}^\dagger]_1, [\mathbf{y}_1^\dagger]_1, [\mathbf{y}_2^\dagger]_2), \mathbf{w}, ([\boldsymbol{\pi}^\dagger]_1, [\boldsymbol{\theta}^\dagger]_2)\right)$.

Note that the commitment keys are perfectly binding at S. First, we claim that in this case the values ρ, srs output by $\mathcal{B}_{\mathbf{S}}$ are identically distributed to honestly computed ones and thus do not skew the probability that $\mathcal{A}$ outputs a valid proof. For ρ this is immediate by the witness samplability of the distributions $\mathcal{M}$, $\mathcal{N}_1$, $\mathcal{N}_2$. We show that this holds for srs as well.

Let $\mathbf{K}_0^\dagger \in \mathbb{F}^{|S_1| \times k}, \mathbf{K}_1^\dagger \in \mathbb{F}^{|S_2| \times k}, \mathbf{K}_2^\dagger \in \mathbb{F}^{|S_2| \times k}$ be the implicit values used to compute $\mathsf{srs}^\dagger$, that is, they satisfy

$$\mathbf{B}^\dagger = \mathbf{M}_S^\top \mathbf{K}_0^\dagger + \mathbf{N}_{1,S}^\top \mathbf{K}_1^\dagger,\ \mathbf{D}^\dagger = \mathbf{N}_{2,S}^\top \mathbf{K}_2^\dagger,\ \mathbf{C}_1^\dagger = \begin{pmatrix} \mathbf{K}_0^\dagger \\ \mathbf{K}_1^\dagger \end{pmatrix} \mathbf{A} \text{ and } \mathbf{C}_2^\dagger = \mathbf{K}_2^\dagger \mathbf{A}.$$

Now $\mathcal{B}_{\mathbf{S}}$ implicitly defines $\mathbf{K}_0 = \mathbf{T_G}\mathbf{K}_0^\dagger + \mathbf{R}_0, \mathbf{K}_1 = \mathbf{T_H}\mathbf{K}_1^\dagger + \mathbf{R}_1, \mathbf{K}_2 = \mathbf{T_F}\mathbf{K}_2^\dagger + \mathbf{R}_2$. First, note that these matrices are uniformly distributed since $\mathbf{R}_0, \mathbf{R}_1, \mathbf{R}_2$ are uniformly distributed. Thus $\mathbf{K}_0, \mathbf{K}_1, \mathbf{K}_2$ are distributed identically to honestly generated values for generating an srs. We claim that the srs output by $\mathcal{A}$ is identically distributed to sampling this matrix and computing the other values honestly. Indeed we have that

$$\begin{aligned}
\mathbf{B} &= \mathbf{B}^\dagger + \mathbf{M}^\top \mathbf{G}^\top \mathbf{R}_0 + \mathbf{N}_1^\top \mathbf{H}^\top \mathbf{R}_1 \\
&= \mathbf{M}_{S_1}^\top \mathbf{K}_0^\dagger + \mathbf{N}_{1,S_2}^\top \mathbf{K}_1^\dagger + \mathbf{M}^\top \mathbf{G}^\top \mathbf{R}_0 + \mathbf{N}_1^\top \mathbf{H}^\top \mathbf{R}_1 \\
&= \mathbf{M}^\top \mathbf{G}^\top \mathbf{T_G} \mathbf{K}_0^\dagger + \mathbf{N}_1^\top \mathbf{H}^\top \mathbf{T_H}^\top \mathbf{K}_1^\dagger + \mathbf{M}^\top \mathbf{G}^\top \mathbf{R}_0 + \mathbf{N}_1^\top \mathbf{H}^\top \mathbf{R}_1 \\
&= \mathbf{M}^\top \mathbf{G}^\top (\mathbf{T_G} \mathbf{K}_0^\dagger + \mathbf{R}_0) + \mathbf{N}_1^\top \mathbf{H}^\top (\mathbf{T_H}^\top \mathbf{K}_1^\dagger + \mathbf{R}_1) \\
&= \mathbf{M}^\top \mathbf{G}^\top \mathbf{K}_0 + \mathbf{N}_1^\top \mathbf{H}^\top \mathbf{K}_1
\end{aligned}$$

where the third equality follows since by the local knowledge soundness of the SSBs we have that $\mathbf{T_G}^\top \mathbf{G} \mathbf{M} = \mathbf{M}_{S_1}$, $\mathbf{T_H}^\top \mathbf{H} \mathbf{N}_1 = \mathbf{N}_{1,S_2}$. Similarly, we have

$$\begin{aligned}
\mathbf{D} &= \mathbf{D}^\dagger + \mathbf{N}_2^\top \mathbf{F}^\top \mathbf{R}_2 \\
&= \mathbf{N}_{2,S_2}^\top \mathbf{K}_2^\dagger + \mathbf{N}_2^\top \mathbf{F}^\top \mathbf{R}_2 \\
&= \mathbf{N}_2^\top \mathbf{F}^\top \mathbf{T_F} \mathbf{K}_2^\dagger + \mathbf{N}_2^\top \mathbf{F}^\top \mathbf{R}_2 \\
&= \mathbf{N}_2^\top \mathbf{F}^\top (\mathbf{T_F} \mathbf{K}_2^\dagger + \mathbf{R}_2) \\
&= \mathbf{N}_2^\top \mathbf{F}^\top \mathbf{K}_2
\end{aligned}$$

Also, we have that

$$\mathbf{C}_1 = \begin{pmatrix} \mathbf{T_G} & \mathbf{0} \\ \mathbf{0} & \mathbf{T_H} \end{pmatrix} \mathbf{C}_1^\dagger + \begin{pmatrix} \mathbf{R}_0 \\ \mathbf{R}_1 \end{pmatrix} \mathbf{A} = \begin{pmatrix} \mathbf{T_G} & \mathbf{0} \\ \mathbf{0} & \mathbf{T_H} \end{pmatrix} \begin{pmatrix} \mathbf{K}_1^\dagger \\ \mathbf{K}_2^\dagger \end{pmatrix} \mathbf{A} + \begin{pmatrix} \mathbf{R}_0 \\ \mathbf{R}_1 \end{pmatrix} \mathbf{A} = \begin{pmatrix} \mathbf{T_G} \mathbf{K}_1^\dagger + \mathbf{R}_0 \\ \mathbf{T_H} \mathbf{K}_2^\dagger + \mathbf{R}_1 \end{pmatrix} \mathbf{A} = \begin{pmatrix} \mathbf{K}_1 \\ \mathbf{K}_2 \end{pmatrix} \mathbf{A}$$

$$\mathbf{C}_2 = \mathbf{T_F} \mathbf{C}_2^\dagger + \mathbf{R}_2 \mathbf{A} = \mathbf{T_F} \mathbf{K}_2^\dagger \mathbf{A} + \mathbf{R}_2 \mathbf{A} = (\mathbf{T_F} \mathbf{K}_2^\dagger + \mathbf{R}_2) \mathbf{A} = \mathbf{K}_2 \mathbf{A}$$

so the outputted srs is indeed identically distributed to an honest one.

Then, we show that $\mathcal{B}$ outputs a valid statement-proof pair w.r.t. to $\mathsf{srs}^\dagger$. Since the commitment keys are extractable and perfectly binding at S, we have that $\mathbf{x}^\dagger$, $\mathbf{y}_1^\dagger$ and $\mathbf{y}_2^\dagger$ are valid openings for the commitments given. Assuming $\mathcal{A}$ produces a valid statement for $\mathcal{R}^{\mathsf{no}}_{\rho,\mathbf{S}}$, for the extracted values it holds that $\mathbf{x}^\dagger = \mathbf{M}_{S_1}\mathbf{w}$ and either $\mathbf{y}_1^\dagger \neq \mathbf{N}_{1,S_2}\mathbf{w}$ or $\mathbf{y}_2^\dagger \neq \mathbf{N}_{2,S_2}\mathbf{w}$. Thus, $\mathcal{B}_{\mathbf{S}}$ outputs a valid statement and it suffices to show that $[\boldsymbol{\pi}^\dagger]_1, [\boldsymbol{\theta}^\dagger]_2$ is a valid proof. Indeed, we have that

$$
\begin{aligned}
\mathbf{0} &= \boldsymbol{\pi}\mathbf{A} + \boldsymbol{\theta}\mathbf{A} - (\mathbf{c}^\top \mid \mathbf{d}_1^\top)\mathbf{C}_1 - \mathbf{d}_2^\top\mathbf{C}_2 \\
&= (\boldsymbol{\pi}^\dagger + \mathbf{c}^\top\mathbf{R}_0 + \mathbf{d}_1^\top\mathbf{R}_1)\mathbf{A} + (\boldsymbol{\theta}^\dagger + \mathbf{d}_2^\top\mathbf{R}_2)\mathbf{A} \\
&\quad - (\mathbf{c}^\top \mid \mathbf{d}_1^\top)\left(\begin{pmatrix}\mathbf{T_G} & \mathbf{0} \\ \mathbf{0} & \mathbf{T_H}\end{pmatrix}\mathbf{C}_1^\dagger + \begin{pmatrix}\mathbf{R}_0 \\ \mathbf{R}_1\end{pmatrix}\mathbf{A}\right) \\
&\quad - \mathbf{d}_2^\top\left(\mathbf{T_F}\mathbf{C}_2^\dagger + \mathbf{R}_2\mathbf{A}\right) \\
&= (\boldsymbol{\pi}^\dagger + \mathbf{c}^\top\mathbf{R}_0 + \mathbf{d}_1^\top\mathbf{R}_2)\mathbf{A} + (\boldsymbol{\theta}^\dagger + \mathbf{d}_2^\top\mathbf{R}_2)\mathbf{A} \\
&\quad - (\mathbf{c}^\top\mathbf{T_G} \mid \mathbf{d}_1^\top\mathbf{T_H})\mathbf{C}_1^\dagger - (\mathbf{c}^\top\mathbf{R}_0 - \mathbf{d}_1^\top\mathbf{R}_1)\mathbf{A} \\
&\quad - \mathbf{d}_2^\top\mathbf{T_F}\mathbf{C}_2^\dagger - \mathbf{d}_2^\top\mathbf{R}_2\mathbf{A} \\
&= \boldsymbol{\pi}^\dagger\mathbf{A} + \boldsymbol{\theta}^\dagger\mathbf{A} - (\mathbf{c}^\top\mathbf{T_G} \mid \mathbf{d}_1^\top\mathbf{T_H})\mathbf{C}_1^\dagger - \mathbf{d}_2^\top\mathbf{T_F}\mathbf{C}_2^\dagger \\
&= \boldsymbol{\pi}^\dagger\mathbf{A} + \boldsymbol{\theta}^\dagger\mathbf{A} - (\mathbf{x}^{\dagger\top} \mid \mathbf{y}_1^{\dagger\top})\mathbf{C}_1^\dagger - \mathbf{y}_2^{\dagger\top}\mathbf{C}_2^\dagger
\end{aligned}
$$

and the last equation is the verification equation for the knowledge transfer argument for $\mathsf{srs}^\dagger$. □

We next show that when the distributions $\mathcal{M}, \mathcal{N}_1, \mathcal{N}_2$ guarantee that the linear knowledge transfer argument is secure w.r.t. all possible sets $\mathbf{S}$, construction QABLin has local knowledge soundness.

We also consider some specific distributions we will use later, namely distributions that extend other distributions with some 0 columns.

Corollary 1. *Let $\mathcal{D}_k$ be a matrix distribution for which $\mathcal{D}_k$-SKerMDH and $\mathbf{M}, \mathbf{N}_1, \mathbf{N}_2$. Then If $\mathcal{D}_k$-SKerMDH assumption holds and for all $S_1 \subseteq [d]$ with $S_1 \leq K_1$, $\mathcal{M}_{S_1}^\top$-MDDH holds,* QABLin *is a locally extractable proof system. The property is preserved even against adversaries that know the discrete logs $\mathbf{N}_1, \mathbf{N}_2$ and the commitment keys on the field.*

Proof. The proof is an immediate consequence of of Thm. 24 and Thm. 9. Note that local knowledge soundness does not depend on assumptions on $\mathbf{N}_1, \mathbf{N}_2$ or the commitments keys, so the property is preserved even when the adversary knows the corresponding discrete logarithms. □

The proof of oblivious trapdoor generation follows from the oblivious trapdoor generation and index set hiding of SSB commitments. We follow essentially the same proof as in the unilateral case.

First we show that we construct an indistinguishable srs given only the commitment keys and the matrices $\mathbf{M}, \mathbf{N}_1, \mathbf{N}_2$.

Lemma 2. *There exists a modified* srs *generation algorithm* K′ *that on input* (ρ, θ')*, where* θ' *contains only either* $\mathbf{M}, \mathbf{N}_1, \mathbf{N}_2$ *or* $\mathbf{G}, \mathbf{H}, \mathbf{F}$ *and outputs an* srs *such that* (ρ, srs) *are identically distributed to the honest algorithm* K*.*

The lemma follows directly by noting that $[\mathbf{B}]_1, [\mathbf{D}]_2$ are efficiently computable given the commitment keys and the discrete logarithms of matrices $\mathbf{M}, \mathbf{N}_1, \mathbf{N}_2$ (equivalently $\mathbf{G}, \mathbf{H}, \mathbf{F}$). As in the unilateral case, we abuse notation and refer to $\mathsf{K}'(\rho, \theta')$ as $\mathsf{K}(\rho, \theta')$.

In the next theorem we consider the three commitment keys issued as a single key. It is easy to verify that the properties of the commitment keys still hold. Essentially, we want to capture the condition that the keys preserve oblivious key generation.

Theorem 25. *Let* $\mathcal{M}$*,* $\mathcal{N}_1$*,* $\mathcal{N}_2$ *be witness samplable distributions, and* CS *be an algebraic SSB commitment scheme and let* CS′ *be the concatenation of three instances of* CS*, that is it outputs* $\mathbf{G}^* = \begin{pmatrix} [\mathbf{G}_0]_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & [\mathbf{G}_1]_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & [\mathbf{G}_2]_2 \end{pmatrix}$ *with* $\mathbf{G}_i \leftarrow \mathsf{CS.KeyGen}(\mathsf{gk}, n, d, K_i, S_i)$*. If* CS′ *is oblivious, then construction* QABLin *of Fig. 4.6 is also oblivious.*

Proof. Since the commitment key is perfectly binding at the extraction set, it is enough to show that index set hiding holds and that we can sample a tuple (ρ, srs) indistinguishable from the one we are given, along with a valid trapdoor.

For index set hiding, it is enough to notice that the srs of QABLin can be efficiently computed given only $[\mathbf{G}]_1, [\mathbf{H}]_1, [\mathbf{F}]_2$. Indeed by sampling $\mathbf{M} \leftarrow \mathcal{M}, \mathbf{N}_1 \leftarrow \mathcal{N}_1, \mathbf{N}_2 \leftarrow \mathcal{N}_2$ all values of srs are efficiently computable as noted in Lemma 2. Thus, a distinguishing advantage in index set hiding of QABLin immediately implies equal advantage on the respective property of CS.

For oblivious srs generation we first describe the OblSetup algorithm. Let $\mathbf{S}' \subseteq \mathbf{S}$.

$\mathsf{OblSetup}(\rho := ([\mathbf{G}^*]_1, [\mathbf{M}]_1, [\mathbf{N}_1]_1, [\mathbf{N}_2]_2), \mathsf{srs})$**:**

$([\mathbf{G}^{*\prime}]_1, \mathbf{T}^{*\prime}_{\mathbf{G}}) \leftarrow \mathsf{CS.OblSetup}(\mathsf{gk}, \mathsf{d}, \mathsf{K}_0, (\mathsf{S}'_1, \mathsf{S}'_2), [\mathbf{G}^*]_1)$
Sample $([\mathbf{M}']_1, \mathbf{M}') \leftarrow \mathcal{N}$; $([\mathbf{N}'_1]_1, \mathbf{N}'_1) \leftarrow \mathcal{N}_2$; $([\mathbf{N}'_2]_2, \mathbf{N}'_2) \leftarrow \mathcal{N}_2$
Set $\tau' = (\mathbf{T}'_{\mathbf{G}}, \mathbf{T}'_{\mathbf{H}}, \mathbf{T}'_{\mathbf{F}})$ and compute $\mathsf{srs} \leftarrow \mathsf{QABLin.K}(\rho, \theta' = (\mathbf{M}, \mathbf{N}_1, \mathbf{N}_2))$

Note that the only difference in sampling with $\mathbf{S}$ and with $\mathbf{S}'$ is how we sample the commitment keys $\mathbf{G}, \mathbf{H}, \mathbf{F}$; srs is identically distributed to an honest one since we sample $\mathbf{M}, \mathbf{N}_1, \mathbf{N}_2$

in the same way that $\mathcal{D}_{\mathsf{par}}$ does. Also, by oblivious key generation of CS, the trapdoor τ' is a valid one w.r.t. $\mathbf{G}', \mathbf{H}', \mathbf{F}'$ and set $\mathbf{S}'$, so it extracts valid witnesses which, by statistical binding in $\mathbf{S}'$ are unique and do not assist the distinguisher which can compute them itself. □

Finally, we show that if we use MPed of Fig. 4.3 for the former scheme and KMPed of Fig. 4.4 for the other two, construction QABLin is no-signaling.

Corollary 2. *Let* CS *be the construction of Fig. 4.3 and* CS′ *be the concatenation of the instances of it. If* $\mathsf{ck} = ([\mathbf{G}]_1, [\mathbf{H}]_1, [\mathbf{F}]_1)$ *is a commitment key from* CS′ *and* $\mathbf{Z}$ *a uniformly distributed matrix, then construction* QABLin *is no-signaling where even when the adversary also knows values* $[\mathbf{H} \otimes \mathbf{F} - \mathbf{Z}]_1, [\mathbf{Z}]_2, \mathbf{M}, \mathbf{N}_1, \mathbf{N}_2$.

Proof. Follows directly from Thm. 15, the oblivious property of the QALin which was shown on Thm. 25 and the properties of the KMPed commitment scheme (Thm. 14). Note that the no-signaling property does not depend on assumptions on $\mathbf{M}, \mathbf{N}_1, \mathbf{N}_2$, so the property is preserved even when the adversary knows the corresponding discrete logarithms. □

4.7.2 Security Analysis of QASum

Theorem 26. *Let* $\mathcal{M}_1, \mathcal{M}_2$ *be (possibly correlated) witness samplable distributions,* $\mathcal{N}$ *be a witness samplable distribution,* $\mathcal{D}_k$ *a matrix distribution and* CS, CS′ *an algebraic and split algebraic SSB commitment respectively. Then,* QASum *is complete and its local knowledge soundness reduces to knowledge transfer soundness of* KTSum.

Proof. For completeness, we have that

$$\begin{aligned}
(\mathbf{c}_1^\top \mid \mathbf{d}^\top)\mathbf{C}_1 + \mathbf{c}_2^\top \mathbf{C}_2 &= (\mathbf{c}_1^\top \mid \mathbf{d}^\top) \begin{pmatrix} \mathbf{K}_0 \\ \mathbf{K}_1 \end{pmatrix} \mathbf{A} + \mathbf{c}_2^\top \mathbf{K}_0 \mathbf{A} \\
&= (\mathbf{c}_1^\top \mathbf{K}_0 + \mathbf{d}^\top \mathbf{K}_1 + \mathbf{c}_2^\top \mathbf{K}_0)\mathbf{A} \\
&= \left((\mathbf{c}_1^\top + \mathbf{c}_2^\top)\mathbf{K}_0 + \mathbf{d}^\top \mathbf{K}_1\right) \mathbf{A} \\
&= \left(\mathbf{w}^\top (\mathbf{M}_1^\top + \mathbf{M}_2^\top)(\mathbf{Q}_1^\top + \mathbf{Q}_2^\top)\mathbf{K}_0 + \mathbf{w}^\top \mathbf{N}^\top \mathbf{F}^\top \mathbf{K}_1\right) \mathbf{A} \\
&= \mathbf{w}^\top \left((\mathbf{M}_1^\top + \mathbf{M}_2^\top)\mathbf{Q}^\top \mathbf{K}_0 + \mathbf{N}^\top \mathbf{F}^\top \mathbf{K}_1\right) \mathbf{A} \\
&= \mathbf{w}^\top \left((\mathbf{M}_1^\top \mathbf{Q}^\top \mathbf{K}_0 + \mathbf{N}^\top \mathbf{F}^\top \mathbf{K}_1 + \mathbf{Z}) + (\mathbf{M}_2^\top \mathbf{Q}^\top \mathbf{K}_0 - \mathbf{Z})\right) \mathbf{A} \\
&= \mathbf{w}^\top (\mathbf{B} + \mathbf{D})\, \mathbf{A} \\
&= \mathbf{w}^\top \mathbf{B}\mathbf{A} + \mathbf{w}^\top \mathbf{D}\mathbf{A} \\
&= \boldsymbol{\pi}\mathbf{A} + \boldsymbol{\theta}\mathbf{A}
\end{aligned}$$

Local knowledge soundness follows using almost an identical argument to Thm. 24 and reducing to knowledge transfer of KTA Sum Argument KW-KTSum of Fig. 4.1. Given an

adversary $\mathcal{A}$ breaking Knowledge Transfer of the quasi-argument of Fig. 4.7, we construct another adversary $\mathcal{B}_{\mathbf{S}}$ that breaks Knowledge Transfer of the argument KW-KTSum for matrices $[\mathbf{M}_{1,S_0}]_1, [\mathbf{M}_{2,S_0}]_2$ and $[\mathbf{N}_{S_1}]_1$. $\mathcal{B}_{\mathbf{S}}$ works as follows: it takes input $(\rho^\dagger, \mathsf{srs}^\dagger)$ where

$$\rho^\dagger := (\mathsf{gk}, [\mathbf{M}_{1,S_0}]_1, [\mathbf{M}_{2,S_0}]_2, [\mathbf{N}_{S_1}]_1),$$
$$\mathsf{srs}^\dagger := ([\mathbf{B}^\dagger]_1, [\mathbf{D}^\dagger]_2, [\mathbf{A}]_{1,2}, [\mathbf{C}_1^\dagger]_2, [\mathbf{C}_2^\dagger]_1)$$

and does the following:

- It samples $([\mathbf{Q}_1]_1, [\mathbf{Q}_2]_2, \mathbf{Q}_1, \mathbf{Q}_2, \mathbf{T}_\mathbf{Q}) \leftarrow \mathsf{CS}'.\mathsf{KeyGen}(gk, d, K, S_1)$ and sets $\mathbf{Q} := \mathbf{Q}_1 + \mathbf{Q}_2$.
- It samples $([\mathbf{F}]_1, \mathbf{F}, \mathbf{T}_\mathbf{F}) \leftarrow \mathsf{CS}.\mathsf{KeyGen}(gk, d, K, S_2)$.
- It samples $\mathbf{M}_{1,\overline{S}_1}, \mathbf{M}_{2,\overline{S}_1}, \mathbf{N}_{\overline{S}_2}$, such that $\mathbf{M}_1 = \mathbf{P}_{S_1}\begin{pmatrix}\mathbf{M}_{1,S_1}\\ \mathbf{M}_{1,\overline{S}_1}\end{pmatrix}$, $\mathbf{M}_2 = \mathbf{P}_{S_1}\begin{pmatrix}\mathbf{M}_{2,S_1}\\ \mathbf{M}_{2,\overline{S}_1}\end{pmatrix}$, $\mathbf{N} = \mathbf{P}_{S_2}\begin{pmatrix}\mathbf{N}_{S_2}\\ \mathbf{N}_{\overline{S}_2}\end{pmatrix}$.
- It samples $\mathbf{R}_0 \leftarrow \mathbb{F}^{\overline{K}_0 \times k}$; $\mathbf{R}_1 \leftarrow \mathbb{F}^{\overline{K}_1 \times k}$.
- It computes
$$[\mathbf{B}]_1 := [\mathbf{B}^\dagger]_1 + [\mathbf{M}_1]_1^\top \mathbf{Q}^\top \mathbf{R}_0 + [\mathbf{N}]_1^\top \mathbf{F}^\top \mathbf{R}_1$$
$$[\mathbf{D}]_2 := [\mathbf{D}^\dagger]_2 + [\mathbf{M}_2]_2^\top \mathbf{Q}^\top \mathbf{R}_0$$
- It computes $[\mathbf{C}_1]_2 := \begin{pmatrix}\mathbf{T}_\mathbf{Q} & \mathbf{0}\\ \mathbf{0} & \mathbf{T}_\mathbf{F}\end{pmatrix}[\mathbf{C}_1^\dagger]_2 + \begin{pmatrix}\mathbf{R}_0\\ \mathbf{R}_1\end{pmatrix}[\mathbf{A}]_2$ and $[\mathbf{C}_2]_1 := \mathbf{T}_\mathbf{Q}[\mathbf{C}_2^\dagger]_1 + \mathbf{R}_0[\mathbf{A}]_1$.
- It sets
$$\rho := ([\mathbf{Q}_1]_1, [\mathbf{Q}_2]_1, [\mathbf{F}]_2, [\mathbf{M}_1]_1, [\mathbf{M}_2]_2, [\mathbf{N}]_1), \quad h_{\mathsf{ls}}(\theta) := (\mathbf{Q}_1, \mathbf{Q}_2, \mathbf{F}, \mathbf{N})$$
$$\mathsf{srs} := ([\mathbf{B}]_1, [\mathbf{D}]_2, [\mathbf{A}]_{1,2}, [\mathbf{C}_1]_2, [\mathbf{C}_2]_1)$$

It then executes $\mathcal{A}(\rho, \mathsf{srs})$ until it outputs a statement $([\mathbf{c}_1]_1, [\mathbf{c}_2]_2, [\mathbf{d}]_1, \mathbf{w})$ together with an accepting proof $[\boldsymbol{\pi}]_1, [\boldsymbol{\theta}]_2$. Given an accepting proof $\mathcal{B}$ sets

$$[\mathbf{x}_1^\dagger]_1 = \mathbf{T}_\mathbf{Q}[\mathbf{c}_1]_1, [\mathbf{x}_2^\dagger]_2 = \mathbf{T}_\mathbf{Q}[\mathbf{c}_2]_2, [\mathbf{y}^\dagger]_1 = \mathbf{T}_\mathbf{F}[\mathbf{d}]_1$$
$$[\boldsymbol{\pi}^\dagger]_1 = [\boldsymbol{\pi}]_1 - [\mathbf{c}_1]_1^\top \mathbf{R}_1 - [\mathbf{d}]_1^\top \mathbf{R}_2, \qquad [\boldsymbol{\theta}^\dagger]_2 = [\boldsymbol{\theta}]_1 - [\mathbf{c}_2]_2^\top \mathbf{R}_1$$

It outputs $\big(([\mathbf{x}_1^\dagger]_1, [\mathbf{x}_2^\dagger]_2, [\mathbf{y}^\dagger]_1), \mathbf{w}, ([\boldsymbol{\pi}^\dagger]_1, [\boldsymbol{\theta}^\dagger]_2)\big)$.

Note that by perfect completeness of the commitment scheme, the commitment keys are extractable and perfectly binding at S.

First, we claim that in this case the values ρ, srs output by $\mathcal{B}_{\mathbf{S}}$ are identically distributed to honestly computed ones and thus do not skew the probability that $\mathcal{A}$ outputs a valid

proof. For ρ, this is immediate by the witness samplability of the distributions $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{N}$. We show that this holds for srs as well. Let $\mathbf{K}_0^\dagger \in \mathbb{F}^{|S_1|\times k}, \mathbf{K}_1^\dagger \in \mathbb{F}^{|S_2|\times k}$, $\mathbf{Z}^\dagger \in \mathbb{F}^{n\times k}$ matrices satisfying:

$$\mathbf{B}^\dagger = \mathbf{M}_{1,S_1}^\top \mathbf{K}_0^\dagger + \mathbf{N}_S^\top \mathbf{K}_1^\dagger + \mathbf{Z}^\dagger,\ \mathbf{D}^\dagger = \mathbf{M}_{2,S_1}^\top \mathbf{K}_0^\dagger - \mathbf{Z}^\dagger,\ \mathbf{C}_1^\dagger = \begin{pmatrix}\mathbf{K}_0^\dagger \\ \mathbf{K}_1^\dagger\end{pmatrix}\mathbf{A} \text{ and } \mathbf{C}_2^\dagger = \mathbf{K}_0^\dagger \mathbf{A}$$

Now $\mathcal{B}_S$ implicitly defines $\mathbf{K}_0 = \mathbf{T}_\mathbf{Q}\mathbf{K}_0^\dagger + \mathbf{R}_0$, $\mathbf{K}_1 = \mathbf{T}_\mathbf{F}\mathbf{K}_1^\dagger + \mathbf{R}_1$, and note that these matrices are uniformly distributed since $\mathbf{R}_0, \mathbf{R}_1$ are uniformly distributed. Thus $\mathbf{K}_0, \mathbf{K}_1$ are distributed identically to honestly generated values for generating an srs. We claim that the srs output by $\mathcal{A}$ is identically distributed to sampling this matrix and computing the other values honestly. Indeed we have that

$$\begin{aligned}
\mathbf{B} &= \mathbf{B}^\dagger + \mathbf{M}_1^\top \mathbf{Q}^\top \mathbf{R}_1 + \mathbf{N}_1^\top \mathbf{H}^\top \mathbf{R}_1 \\
&= \mathbf{M}_{1,S_1}^\top \mathbf{K}_0^\dagger + \mathbf{N}_{S_2}^\top \mathbf{K}_1 + \mathbf{Z}^\dagger + \mathbf{M}_1^\top \mathbf{Q}^\top \mathbf{R}_0 + \mathbf{N}_1^\top \mathbf{H}^\top \mathbf{R}_1 \\
&= \mathbf{M}_1^\top \mathbf{Q}^\top \mathbf{T}_\mathbf{Q} \mathbf{K}_0^\dagger + \mathbf{N}^\top \mathbf{F}^\top \mathbf{T}_\mathbf{F}^\top \mathbf{K}_1 + \mathbf{Z}^\dagger + \mathbf{M}_1^\top \mathbf{Q}^\top \mathbf{R}_0 + \mathbf{N}_1^\top \mathbf{H}^\top \mathbf{R}_1 \\
&= \mathbf{M}_1^\top \mathbf{Q}^\top (\mathbf{T}_\mathbf{Q} \mathbf{K}_0^\dagger + \mathbf{R}_0) + \mathbf{N}^\top \mathbf{F}^\top (\mathbf{T}_\mathbf{F}^\top \mathbf{K}_1 + \mathbf{R}_1) + \mathbf{Z}^\dagger \\
&= \mathbf{M}_1^\top \mathbf{Q}^\top \mathbf{K}_0 + \mathbf{N}^\top \mathbf{F}^\top \mathbf{K}_1 + \mathbf{Z}^\dagger
\end{aligned}$$

where the third equality follows since by the local extractability of the SSBs (1) $\mathbf{T}_\mathbf{Q}^\top \mathbf{Q} \mathbf{M}_1 = \mathbf{M}_{1,S}$ and (2) $\mathbf{T}_\mathbf{F}^\top \mathbf{F} \mathbf{N} = \mathbf{N}_S$. Similarly, we have

$$\begin{aligned}
\mathbf{D} &= \mathbf{D}^\dagger + \mathbf{M}_2^\top \mathbf{Q}^\top \mathbf{R}_0 \\
&= \mathbf{M}_{2,S_1}^\top \mathbf{K}_0^\dagger - \mathbf{Z}^\dagger + \mathbf{M}_2^\top \mathbf{Q}^\top \mathbf{R}_0 \\
&= \mathbf{M}_2^\top \mathbf{Q}^\top \mathbf{T}_\mathbf{Q} \mathbf{K}_0^\dagger - \mathbf{Z}^\dagger + \mathbf{M}_2^\top \mathbf{Q}^\top \mathbf{R}_0 \\
&= \mathbf{M}_2^\top \mathbf{Q}^\top (\mathbf{T}_\mathbf{Q} \mathbf{K}_0^\dagger + \mathbf{R}_0) - \mathbf{Z}^\dagger \\
&= \mathbf{M}_2^\top \mathbf{Q}^\top \mathbf{K}_0 - \mathbf{Z}^\dagger
\end{aligned}$$

Also, we have that

$$\mathbf{C}_1 = \begin{pmatrix}\mathbf{T}_\mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_\mathbf{F}\end{pmatrix}\mathbf{C}_1^\dagger + \begin{pmatrix}\mathbf{R}_0 \\ \mathbf{R}_1\end{pmatrix}\mathbf{A} = \begin{pmatrix}\mathbf{T}_\mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_\mathbf{F}\end{pmatrix}\begin{pmatrix}\mathbf{K}_0^\dagger \\ \mathbf{K}_1^\dagger\end{pmatrix}\mathbf{A} + \begin{pmatrix}\mathbf{R}_0 \\ \mathbf{R}_1\end{pmatrix}\mathbf{A} = i\begin{pmatrix}\mathbf{T}_\mathbf{Q}\mathbf{K}_0^\dagger + \mathbf{R}_0 \\ \mathbf{T}_\mathbf{F}\mathbf{K}_1^\dagger + \mathbf{R}_1\end{pmatrix}\mathbf{A} = \begin{pmatrix}\mathbf{K}_0 \\ \mathbf{K}_1\end{pmatrix}\mathbf{A}$$

$$\mathbf{C}_2 = \mathbf{T}_\mathbf{Q}\mathbf{C}_2^\dagger + \mathbf{R}_0\mathbf{A} = \mathbf{T}_\mathbf{Q}\mathbf{K}_0^\dagger\mathbf{A} + \mathbf{R}_0\mathbf{A} = (\mathbf{T}_\mathbf{Q}\mathbf{K}_0^\dagger + \mathbf{R}_0)\mathbf{A} = \mathbf{K}_0\mathbf{A}$$

so the outputted srs is indeed identically distributed to an honest one.

Then, we show that $\mathcal{B}$ outputs a valid statement-proof pair w.r.t. to $\mathsf{srs}^\dagger$. Since the commitment keys are extractable and perfectly binding, we have that $(\mathbf{x}_1^\dagger, \mathbf{x}_2^\dagger)$ and $\mathbf{y}^\dagger$ are valid openings for the commitments $(\mathbf{c}_1, \mathbf{c}_2)$ and $\mathbf{d}$ respectively. Assuming $\mathcal{A}$ produces a valid statement for $\mathsf{KTSum}\text{-}\mathcal{R}_{\rho,\mathbf{S}}^{\mathsf{no}}$, for the extracted values it holds that $\mathbf{x}_1^\dagger + \mathbf{x}_2^\dagger = (\mathbf{M}_{1,S_1} + \mathbf{M}_{2,S_1})\mathbf{w}$ and $\mathbf{y}^\dagger \neq \mathbf{N}_{S_2}\mathbf{w}$. Thus, $\mathcal{B}_\mathbf{S}$ outputs a valid statement and it suffices to show that $(\boldsymbol{\pi}^\dagger, \boldsymbol{\theta}^\dagger)$ is

a valid proof. Indeed, we have

$$\begin{aligned}
\mathbf{0} &= \boldsymbol{\pi}\mathbf{A} + \boldsymbol{\theta}\mathbf{A} - (\mathbf{c}_1^\top \mid \mathbf{d}^\top)\mathbf{C}_1 - \mathbf{c}_2^\top \mathbf{C}_2 \\
&= (\boldsymbol{\pi}^\dagger + \mathbf{c}_1^\top \mathbf{R}_0 + \mathbf{d}^\top \mathbf{R}_1)\mathbf{A} + (\boldsymbol{\theta}^\dagger + \mathbf{c}_2^\top \mathbf{R}_0)\mathbf{A} \\
&\quad - (\mathbf{c}_1^\top \mid \mathbf{d}^\top)\left(\begin{pmatrix} \mathbf{T_Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{T_F} \end{pmatrix}\mathbf{C}_1^\dagger + \begin{pmatrix} \mathbf{R}_0 \\ \mathbf{R}_1 \end{pmatrix}\mathbf{A}\right) \\
&\quad - \mathbf{c}_2^\top\left(\mathbf{T_Q}\mathbf{C}_2^\dagger + \mathbf{R}_0\mathbf{A}\right) \\
&= (\boldsymbol{\pi}^\dagger + \mathbf{c}_1^\top \mathbf{R}_0 + \mathbf{d}^\top \mathbf{R}_1)\mathbf{A} + (\boldsymbol{\theta}^\dagger + \mathbf{c}_2^\top \mathbf{R}_0)\mathbf{A} \\
&\quad - (\mathbf{c}_1^\top \mathbf{T_Q} \mid \mathbf{d}^\top \mathbf{T_F})\mathbf{C}_1^\dagger - (\mathbf{c}_1^\top \mathbf{R}_0 - \mathbf{d}^\top \mathbf{R}_1)\mathbf{A} \\
&\quad - \mathbf{c}_2^\top \mathbf{T_Q}\mathbf{C}_2^\dagger - \mathbf{c}_2^\top \mathbf{R}_0\mathbf{A} \\
&= \boldsymbol{\pi}^\dagger\mathbf{A} + \boldsymbol{\theta}^\dagger\mathbf{A} - (\mathbf{c}_1^\top \mathbf{T_Q} \mid \mathbf{d}^\top \mathbf{T_F})\mathbf{C}_1^\dagger - \mathbf{c}_2^\top \mathbf{T_Q}\mathbf{C}_2^\dagger \\
&= \boldsymbol{\pi}^\dagger\mathbf{A} + \boldsymbol{\theta}^\dagger\mathbf{A} - (\mathbf{x}_1^{\dagger\top} \mid \mathbf{y}^{\dagger\top})\mathbf{C}_1^\dagger - \mathbf{x}_2^{\dagger\top}\mathbf{C}_2^\dagger
\end{aligned}$$

and the last equation is the verifying equation for the knowledge transfer argument for $\mathsf{srs}^\dagger$. □

By the above theorem, when the distributions $(\mathcal{M}_1, \mathcal{M}_2), \mathcal{N}, \mathcal{D}_k$ guarantee that the sum knowledge transfer argument is secure w.r.t. all possible sets defined by a set $\mathbf{S}$, construction QASum has local knowledge soundness. We focus on a specific case that considers $\mathcal{M}_1, \mathcal{M}_2$ derived from a distribution $\mathcal{U} \otimes \mathcal{V}$ that satisfies the $(\mathcal{U} \otimes \mathcal{V})^\top$-KMDDH assumption.

Corollary 3. *Let $\mathcal{D}_k$ be a matrix distribution for which $\mathcal{D}_k$-*SKerMDH *and $(\mathcal{M}_1, \mathcal{M}_2)$ be derived from the splitting of $\mathcal{U} \otimes \mathcal{V}$. If for all $S_0 \subseteq [d]$ with $S_0 \leq K_0$, $(\mathcal{U} \otimes \mathcal{V})^\top_{1,S_0}$-*KMDDH *holds,* QASum *is a locally extractable proof system. The property is preserved even against adversaries that know the discrete logs $\mathbf{N}$ and the commitment keys on the field.*

Proof. The proof is an immediate consequence of Thm. 26 and Thm. 9. Local knowledge soundness does not depend on assumptions on $\mathbf{N}$ or the commitments keys, so the property is preserved even when the adversary knows the corresponding discrete logarithms. □

The proof that QASum is oblivious follows from the oblivious trapdoor generation and index set hiding of the SSB commitments. We essentially follow the same proof as in the QABLin case.

First we show the corresponding lemma to Lemma 2, that is, we construct an indistinguishable srs given only the commitment keys and the matrices $\mathbf{M}_1, \mathbf{M}_2, \mathbf{N}$.

Lemma 3. *There exists a modified* srs *generation algorithm* K′ *that on input (ρ, θ'), where θ' contains only either $\mathbf{M}_1, \mathbf{M}_2, \mathbf{N}$ or $\mathbf{Q}_1, \mathbf{Q}_2, \mathbf{F}$ and outputs an* srs *such that (ρ, srs) are identically distributed to the honest algorithm.*

Proof. Given these values we can compute the srs using a simple trick. Instead of computing

$$[\mathbf{B}]_1 = [\mathbf{M}_1^\top]_1 \mathbf{Q}^\top \mathbf{K}_0 + [\mathbf{N}^\top]_1 \mathbf{F}^\top \mathbf{K}_1 + [\mathbf{Z}]_1$$
$$[\mathbf{D}]_2 = [\mathbf{M}_2^\top]_2 \mathbf{Q}^\top \mathbf{K}_0 - [\mathbf{Z}]_2,$$

we compute

$$[\mathbf{B}]_1 = (\mathbf{M}_1^\top + \mathbf{M}_2^\top)[\mathbf{Q}_1^\top]_1 \mathbf{K}_0 + [\mathbf{N}^\top]_1 \mathbf{F}^\top \mathbf{K}_1 + [\mathbf{Z}]_1$$
$$[\mathbf{D}]_2 = (\mathbf{M}_1^\top + \mathbf{M}_2^\top)[\mathbf{Q}_2^\top]_2 \mathbf{K}_0 - [\mathbf{Z}]_2$$

Noting that in both cases the elements computed are uniformly distributed conditioned on $\mathbf{B} + \mathbf{D} = (\mathbf{M}_1^\top + \mathbf{M}_2^\top)(\mathbf{Q}_1^\top + \mathbf{Q}_2^\top)\mathbf{K}_0 + \mathbf{N}^\top \mathbf{F}^\top \mathbf{K}_1$ we see that these values are computed as in the honest setup.

In the case where $\theta = (\mathbf{Q}_1, \mathbf{Q}_2, \mathbf{F})$ we can directly compute the srs by noting that $\mathbf{Q} = \mathbf{Q}_1 + \mathbf{Q}_2$ and the group elements in ρ are enough to compute all values of srs. □

As in the previous cases, we abuse notation and refer to $\mathsf{K}'(\rho, \theta')$ as $\mathsf{K}(\rho, \theta')$.

The proof of oblivious extraction essentially follows from the oblivious key generation and index set hiding of the SSB commitments and is similar to the proof of Thm. 25.

Theorem 27. *Let $(\mathcal{M}_1, \mathcal{M}_2), \mathcal{N}, \mathcal{D}_k$ be witness samplable distribution, and* CS, CS′ *be an algebraic and a split algebraic SSB commitment scheme. If* CS, CS′ *are oblivious then* QASum *is also oblivious.*

Proof. It is enough to show that index set hiding holds and that we can sample a tuple (ρ, srs) indistinguishable from the one we are given, along with a valid trapdoor. This is the case because the commitment keys are perfectly binding in S', which means that the witnesses are unique and do not help the (unbounded) distinguisher who can compute them on its own.

Index Set Hidning. Assume there exist sets $\mathbf{S}, \mathbf{S}'$ of size at most $\mathbf{K}$ and an adversary $\mathcal{A}$ which distinguishes (ρ, srs) sampled for $\mathbf{S}$ from (ρ, srs) sampled for $\mathbf{S}'$ with some probability α. We construct adversaries $\mathcal{B}_0$ distinguishing ck_0 sampled for S_0 from ck_0 sampled for S_0' with probability α_0 and an adversary $\mathcal{B}_1$ distinguishing ck_1 sampled for S_1 from ck_1 sampled for S_1' with probability α_1 such that $\alpha \leq \frac{\alpha_0 + \alpha_1}{2}$.

$\mathcal{B}_0$ takes as input ck_0 sampled either for S_0 or S_0'. It then honestly computes the srs by sampling $\mathbf{M}_1, \mathbf{M}_2, \mathbf{N}$ and following the K described in Lemma 3 except that ck_1 is computed as follows: it samples $b \leftarrow \{0, 1\}$ and if $b = 0$ it sets $(\mathsf{ck}_1, sk_1) \leftarrow \mathsf{CS.KeyGen}(\mathsf{gk}_1, d, K, S_1)$ otherwise it sets $(\mathsf{ck}_1, sk_1) \leftarrow \mathsf{CS.KeyGen}(\mathsf{gk}_1, d, K, S_1')$.Note that, with probability $1/2$, the srs computed by $\mathcal{B}$ follows exactly the original distribution. This is the case since $\mathbf{B}, \mathbf{D}$ are uniform matrices conditioned on their sum being equal to $(\mathbf{M}_1^\top + \mathbf{M}_2^\top)(\mathbf{Q}_1^\top + \mathbf{Q}_2^\top)\mathbf{K}_1 + \mathbf{N}^\top \mathbf{F}^\top \mathbf{K}_2$

for uniform $\mathbf{K}_1, \mathbf{K}_2$, exactly as in the honest srs generation. Finally $\mathcal{B}_0$ runs $\mathcal{A}(\rho, \mathsf{srs})$ and output whatever it outputs.

Similarly, on input ck_1 sampled either for S_1 or S_1', $\mathcal{B}_1$ samples $b \leftarrow \{0,1\}$ and if $b = 0$ it sets $(\mathsf{ck}_0, sk_0) \leftarrow \mathsf{CS.KeyGen}(\mathsf{gk}, d, K, S_0)$ otherwise it sets $(\mathsf{ck}_0, sk_0) \leftarrow \mathsf{CS.KeyGen}(\mathsf{gk}, d, K, S_0')$ and honestly computes the srs as in the previous case. A simple case analysis shows that $\alpha \leq \frac{\alpha+\alpha}{2}$.

Oblivious trapdoor generation: We show how to obliviously sample a trapdoor given black box access to CS.OblKeyGen and CS′.OblKeyGen. For oblivious trapdoor generation, given a pair ρ, srs for the quasi argument and set $\mathbf{S}'$ the oblivious setup QASum.OblKeyGen does the following:

- $(\mathsf{ck}_0', \tau_0') \leftarrow \mathsf{CS.OblKeyGen}(\mathsf{ck}_0, S_0')$ and
  $(\mathsf{ck}_1', \tau_1') \leftarrow \mathsf{CS'.OblKeyGen}(\mathsf{ck}_1, S_1')$.
- Sample $([\mathbf{M}_1]_1, [\mathbf{M}_2]_2, \mathbf{M}_1, \mathbf{M}_2) \leftarrow \mathcal{N}$, $([\mathbf{N}]_1, \mathbf{N}) \leftarrow \mathcal{N}$.
- Compute the rest of the srs by $\mathsf{K}(\mathsf{ck}_0', \mathsf{ck}_1', \mathbf{M}_1, \mathbf{M}_2, \mathbf{N})$.

Arguing as in the index set hiding proof, the only difference in the oblivious and an honest srs is how the commitment keys are sampled. We can thus use a standard hybrid argument to reduce the property to the oblivious trapdoor generation of the commitment schemes CS, CS′. □

Finally, we consider the specific case of using the KMPed key of 4.4 for ck_0 and MPed of 4.3 for ck_1, construction QASum is no-signaling.

Corollary 4. *Let* CS *be an instance of* KMPed *and* CS′ *an instance of* MPed. *Then* QASum *is no-signaling. The property is preserved even against adversaries that know the discrete logs* $\mathbf{M}_1, \mathbf{M}_2, \mathbf{M}$.

Proof. The proof follows directly from Thm 15 and the oblivious property of QASum, which in turn follows from applying Thm. 10 and Thm. 14 to Theorem 27. The no-signaling property does not depend on assumptions on $\mathbf{M}_1, \mathbf{M}_2, \mathbf{N}$, so the property is preserved even when the adversary knows the corresponding discrete logarithms. □

4.7.3 Security Analysis of QAHad

Theorem 28. *Let* $\mathcal{U}, \mathcal{V}, \mathcal{W}$ *be witness samplable distributions,* $\mathcal{D}_k$ *be a matrix distribution and* CS *an algebraic SSB commitment scheme. Then,* QAHad *is complete and its local knowledge*

soundness reduces to local knowledge soundness of QASum.

Proof. For completeness, we have that

$$\begin{aligned}\mathbf{u} \otimes \mathbf{v} &= \mathbf{GUa} \otimes \mathbf{GUb} = (\mathbf{G} \otimes \mathbf{H})(\mathbf{U} \otimes \mathbf{V})(\mathbf{a} \otimes \mathbf{b}) = \\ &= (\mathbf{G} \otimes \mathbf{H} - \mathbf{Z} + \mathbf{Z})(\mathbf{U} \otimes \mathbf{V} - \mathbf{R} + \mathbf{R})(\mathbf{a} \otimes \mathbf{b}) = \\ &= (\mathbf{Q}_1 + \mathbf{Q}_2)(\mathbf{M}_1 + \mathbf{M}_2)(\mathbf{a} \otimes \mathbf{b})\end{aligned}$$

and also $\mathbf{c}_1 + \mathbf{c}_2 = (\mathbf{E}_1 + \mathbf{E}_2)(\mathbf{a} \otimes \mathbf{b}) = (\mathbf{Q}_1 + \mathbf{Q}_2)(\mathbf{U} \otimes \mathbf{V})(\mathbf{a} \otimes \mathbf{b}) = \mathbf{u} \otimes \mathbf{v}$, so the pairing test is successful. Finally, noting that $\mathbf{w} = \mathbf{d} = \mathbf{FW}(\mathbf{a} \circ \mathbf{b}) = \mathbf{FWD}(\mathbf{a} \otimes \mathbf{b}) = \mathbf{FN}(\mathbf{a} \otimes \mathbf{b})$, we see that the statement/witness pair $([\mathbf{c}_1]_1, [\mathbf{c}_2]_2, [\mathbf{d}]_1), \mathbf{a} \otimes \mathbf{b}$ is a yes instance of the sum language for parameters ρ_{Sum} and the second condition for verification follows by completeness of QASum.

For local knowledge soundness, it is enough to note that the Kronecker part of the knowledge transfer holds unconditionally, that is, if for some promise $\mathbf{a}, \mathbf{b}$ it holds that $\mathbf{u} = \mathbf{GUa}$ and $\mathbf{v} = \mathbf{HVb}$, then by the verification of the pairing condition, $\mathbf{c}_1 + \mathbf{c}_2 = (\mathbf{Q}_1 + \mathbf{Q}_2)(\mathbf{M}_1 + \mathbf{M}_2)(\mathbf{a} \otimes \mathbf{b})$, so we efficiently construct a promise for the sum language. Now, an accepting proof for the Hadamard language contains an accepting proof for the sum language and we use that to break local soundness of QASum. Details follow.

Let $\mathcal{A}$ be an adversary against local knowledge soundness of QAHad. We construct an adversary $\mathcal{B}$ against local knowledge soundness of QASum. We implicitly use the fact that the soundness of QASum holds even against adversaries who know the discrete logarithms of the commitment keys $\mathbf{Q}_1, \mathbf{Q}_2$. $\mathcal{B}$ takes as input $(\rho_{\mathsf{Sum}}, \mathsf{srs}_{\mathsf{Sum}})$ and works as follows:

- Parse
$$\rho_{\mathsf{Sum}} = \begin{pmatrix} \mathsf{gk}, [\mathbf{Q}_1]_1, [\mathbf{Q}_2]_2, [\mathbf{F}]_1, [\mathbf{M}_1]_1, [\mathbf{M}_2]_2, [\mathbf{N}]_1, \\ \mathsf{aux}_{\mathsf{CS}} = ([\mathbf{G}]_1, [\mathbf{H}]_1), \mathsf{aux}_{\mathcal{N}} = ([\mathbf{U}]_1, [\mathbf{V}]_2), \end{pmatrix}$$
- Set $\rho = (\mathsf{gk}, [\mathbf{G}]_1, [\mathbf{H}]_2, [\mathbf{F}]_2, [\mathbf{U}]_1, [\mathbf{V}]_2, [\mathbf{N}]_1)$.
- Sample $\mathbf{R}' \leftarrow \mathbb{F}^{\overline{K} \times n^2}$ and set $[\mathbf{E}_1]_1 = (\mathbf{Q}_1 + \mathbf{Q}_2)[\mathbf{M}_1]_1 + [\mathbf{R}']_1, [\mathbf{E}_2]_2 = (\mathbf{Q}_1 + \mathbf{Q}_2)[\mathbf{M}_2]_2 - [\mathbf{R}']_2$.

It then executes $\mathcal{A}(\rho, \mathsf{srs} = ([\mathbf{E}_1]_1, [\mathbf{E}_2]_2, \mathsf{srs}_{\mathsf{Sum}}))$ until it outputs a statement $([\mathbf{u}]_1, [\mathbf{v}]_2, [\mathbf{w}]_1, \mathbf{a}, \mathbf{b})$ together with an accepting proof $([\mathbf{c}_1]_1, [\mathbf{c}_2]_2, \pi_{\mathsf{Sum}})$. It outputs the tuple containing the following statement, advice and proof:

$$(([\mathbf{c}_1]_1, [\mathbf{c}_2]_2, [\mathbf{w}]_1), \mathbf{a} \otimes \mathbf{b}, \pi_{\mathsf{Sum}})$$

The srs is identically distributed to an honestly computed one. Indeed the only thing computed differently are the values $[\mathbf{E}_1]_1, [\mathbf{E}_2]_2$, but note that in the reduction they are

distributed uniformly conditioned on $\mathbf{E}_1 + \mathbf{E}_2 = (\mathbf{Q}_1 + \mathbf{Q}_2)(\mathbf{M}_1 + \mathbf{M}_2) = (\mathbf{Q}_1 + \mathbf{Q}_2)(\mathbf{U} \otimes \mathbf{V})$, as in the honest srs generation.

Now, assuming an accepting proof, and a correct promise $\mathbf{a}, \mathbf{b}$ given from $\mathcal{A}$ means that the promise of $\mathcal{B}$ is also correct. Indeed, we have

$$\begin{aligned}\mathbf{c}_1 + \mathbf{c}_2 = \mathbf{u} \otimes \mathbf{v} = \mathbf{GUa} \otimes \mathbf{HVb} = (\mathbf{G} \otimes \mathbf{H})(\mathbf{U} \otimes \mathbf{V})(\mathbf{a} \otimes \mathbf{b}) = \\ = (\mathbf{G} \otimes \mathbf{H} - \mathbf{Z} + \mathbf{Z})(\mathbf{U} \otimes \mathbf{V} - \mathbf{R} + \mathbf{R})(\mathbf{a} \otimes \mathbf{b}) = \\ = (\mathbf{Q}_1 + \mathbf{Q}_2)(\mathbf{M}_1 + \mathbf{M}_2)(\mathbf{a} \otimes \mathbf{b}).\end{aligned}$$

Now let $\mathbf{x}_1 = \mathbf{T}_\mathbf{Q}\mathbf{c}_1$, $\mathbf{x}_2 = \mathbf{T}_\mathbf{Q}\mathbf{c}_2$, $\mathbf{y} = \mathbf{T}_\mathbf{F}\mathbf{w}$ be the extracted values. We have that

$$\begin{aligned}\mathbf{x}_1 + \mathbf{x}_2 = \mathbf{T}_\mathbf{Q}(\mathbf{c}_1 + \mathbf{c}_2) = \mathbf{T}_\mathbf{Q}(\mathbf{Q}_1 + \mathbf{Q}_2)(\mathbf{M}_1 + \mathbf{M}_2)(\mathbf{a} \otimes \mathbf{b}) \\ = (\mathbf{M}_{S,1} + \mathbf{M}_{S,2})(\mathbf{a} \otimes \mathbf{b}).\end{aligned}$$

so indeed the promise is correct. Also assuming that the statement/advice given from $\mathcal{A}$ is a no-instance for the Hadamard language w.r.t. to the set S, then the statement/advice given from $\mathcal{B}$ is a no-instance for the sum language w.r.t. the same set S. Indeed, we have

$$\mathbf{y} \neq \mathbf{W}_\mathbf{S}(\mathbf{a} \circ \mathbf{b}) = \mathbf{W}_\mathbf{S}\mathbf{D}(\mathbf{a} \otimes \mathbf{b}) = \mathbf{N}_\mathbf{S}(\mathbf{a} \otimes \mathbf{b}).$$

So, conditioned on a successful $\mathcal{A}$, $\mathcal{B}$ outputs an instance/advice such that (1) the extractor gets values that satisfy $\mathcal{R}^{\mathsf{no}}_{\rho_{\mathsf{Sum}},\mathbf{S}}$ for ρ_{Sum} and (2) a proof that verifies w.r.t. the instance. □

We next show that when the distributions $\mathcal{U}, \mathcal{V}, \mathcal{W}$ guarantee that the sum knowledge transfer argument is secure w.r.t. all possible sets $\mathbf{S}$, construction QAHad satisfies local knowledge soundness.

Corollary 5. *Let $\mathcal{D}_k$ be a matrix distribution for which $\mathcal{D}_k$-SKerMDH. Denote $\mathcal{X}_S$ the distributions that sample matrices from $\mathcal{U}$ (resp. $\mathcal{V}$, $\mathcal{W}$), and restricts them to rows corresponding to S. Then, if for all $S \subseteq [d]$ with $S \leq K$, $(\mathcal{U} \otimes \mathcal{V})^\top_S$-KMDDH holds,* QAHad *satisfies local knowledge soundness. The property is preserved even against adversaries that know the discrete logs $\mathbf{W}$ and the commitment keys on the field.*

Proof. The proof is an immediate consequence of Thm. 28, Thm. 15 and Cor 3. □

The proof of oblivious trapdoor generation essentially follows from the oblivious trapdoor generation and index set hiding of the SSB commitments and is similar to the corresponding proofs for the other constructions.

First we show the corresponding lemma to Lemma 3, that is, we construct an indistinguishable srs given only the commitment keys and the matrices $\mathbf{U}, \mathbf{V}, \mathbf{W}$.

Lemma 4. *There exists a modified* srs *generation algorithm* K′ *that on input* (ρ, θ')*, where either* $\theta' = (\mathbf{U}, \mathbf{V}, \mathbf{W}, [\mathbf{G} \otimes \mathbf{H} - \mathbf{Z}]_1, [\mathbf{Z}]_2)$ *or* $\theta' = (\mathbf{G}, \mathbf{H}, \mathbf{F}, [\mathbf{U} \otimes \mathbf{V} - \mathbf{R}]_1, [\mathbf{R}]_2)$ *and outputs a* srs *such that* (ρ, srs) *are identically distributed to the honest algorithm.*

The lemma follows by inspection and by noting that with the given values we can compute the srs for the sum as explained in Lemma 3. Again, w.l.o.g. we use the same name for the two algorithms, namely K and differentiate them by their input.

We next show that the construction satisfies oblivious extractability.

Theorem 29. *Let* $\mathcal{U}, \mathcal{V}, \mathcal{W}$ *be witness samplable distributions, and* CS *be instantiated with* MPed *of Fig. 4.3. Then Construction* QAHad *of Fig. 4.8 is oblivious.*

Proof. It is enough to show that index set hiding holds and that we can sample a tuple (ρ, srs) indistinguishable from the one we are given, along with a valid trapdoor. This is the case because the commitment keys are perfectly binding in S', which means that the witnesses are unique and do not help the (unbounded) distinguisher who can compute them on its own.

Index Set Hidning. The property follows by the index set hiding property of KMPed and MPed and the fact that we compute the rest of the values in ρ, srs as described in Lemma 4.

Oblivious trapdoor generation: Here, we can simply use the oblivious trapdoor generation of protocol QASum. The conditions of Cor. 4 are satisfied. It is enough to show that we can compute the srs for the QAHad given a srs for QASum. But this is easy since when given a pair $(\rho_{\mathsf{Sum}}, \mathsf{srs}_{\mathsf{Sum}})$ we execute the oblivious srs algorithm $\mathsf{QASum.OblKeyGen}(\rho, \mathsf{srs}, \mathbf{S} = (S, S))$ as in Lemma 4. □

Corollary 6. *If we instantiate* CS *with* MPed *of 4.3, then* QAHad *from Fig. 4.8 is no-signaling. The property is preserved even against adversaries that know the discrete logs* $\mathbf{U}, \mathbf{V}, \mathbf{W}$*.*

Proof. The proof follows directly from Thm. 29, and Thm. 15 and Cor. 4. □

Chapter 5

Tree Based Vector Commitment Schemes

Part of this chapter is based on a result form the paper "Linear-map Vector Commitments and their Practical Applications", which is joint work with Matteo Campanelli, Anca Nitulescu, Carla Ráfols, Arantxa Zapico.

Vector commitment schemes, introduced in [LY10; CF13], is a cryptographic primitive that allows a party to commit to a vector $\mathbf{v}$ by producing a digest and later reveal some position of the vector in a verifiable manner. This property is called *position binding*. The most basic security guarantee is that it is infeasible to open a commitment at position i to two different values $v_i \neq v_i'$). As far as efficiency is concerned, both the commitment and each individual opening proof should be sublinear in the dimension of the committed vector.

There is a plethora of additional requirements for vector commitments. One natural generalization is the *subvector opening* property [LM19; BBF19]. This states that a prover can open a "bunch" of position in a single proof, instead of sending individual openings for each. Crucially, it should be the case that the size of the subvector proof should also be sublinear in the number of position proven.

Going even further, one can consider opening *functions* of the committed values, namely, if one commits to some vector $\mathbf{v}$, he should later be able to verifiable reveal a value $f(\mathbf{v})$ for some public fuction f. Such constructions were introduced in [LRY16] and are called Functional Vector Commitments. Lai and Malavolta [LM19] introduced *Linear Map Vector Commitments* which is a special case of functional commitments that allow opening linear maps.

Vector commitments are very useful to scale highly decentralized networks of large size and

whose content is dynamic [CPZ18; BBF19; CFG+20; GRWZ20] (such dynamic content can be the state of a blockchain, amount stored on a wallet, the value of a file in a decentralized storage network, etc.). We discuss some of the most prominent applications of VC to motivate and justify their practical importance.

Verifiable Databases. One of the applications that can be significantly improved by Vector Commitments is Verifiable Databases (VDB). In this setting, a client outsources the storage of a database to a server while keeping the ability to access and change some of its records, i.e. query functions of the data and update some of the data and ensure the server does not tamper with the data.

Stateless Cryptocurrency. A recent application that motivated more efficient constructions of VC schemes is *stateless cryptocurrency*, i.e. a payment system based on a distributed ledger where neither validators of transactions nor system users need to store the full ledger state.

Proof of Space. Proof of Space (PoS) is a protocol that allows miners (storage providers) to convince the network that they are dedicating physical storage over time in an efficient way. In a nutshell, a miner commits to a file (data) that uses a specified amount of disk space and then the miner proves that it continues to store the data by answering to recurring audits that consist of random spot-checks.

"Caching" Optimizations. In some applications, e.g. when performing HTTP queries, clients use the so-called *prefetching*[1] and receive from a server not only the values of interest but other related values that could potentially be queried in the near future (e.g., values in a neighboring range of the queried values). Vector commitments with efficient proofs for special ("caching") subset openings allow to add verifiability to such queries in a way that does not affect the speed of the server since the proving procedure for a bigger subset is close or the same as for individual positions.

Improved Succinct Arguments. Construction of succinct arguments has been abstracted and modularized in the recent years. One particular such way is constructing an proof system in an idealized model called Interactive Oracle Proofs [BCS16] and then "compile" it to an actual proof system using Vector Commitments. More efficient VC constructions directly yield more efficient succinct arguments that are constructed in this way.

Properties of Vector Commitments.

Apart from efficiency -which is always a requirement, especially in practical applications- a vector commitment is required to satisfy a plethora of other properties. In a nutshell, a vector commitment scheme should satisfy:

- *Expressivity.* One would like VC to be as expressive as possible, meaning that it should

[1] https://developer.mozilla.org/en-US/docs/Web/HTTP/Link_prefetching_FAQ

be possible to open to functions of the vector as general as possible (subvector openings, linear or arbitrary functions).

- *Privacy.* The commitment should not reveal information about the committed vector. Furthermore, the proofs of claims should not reveal more information apart from the claim itself.

- *Proof Aggregation.* This property captures the ability to "pack" two or more proofs together obtaining a new proof for their combined claims. This should be possible without knowledge of the opening of the vector and aggregation cost should be sublinear in the vector length. Importantly, the resulting proof size should not significantly grow each time we perform an aggregation.

- *Updatability.* This property allows to efficiently update opening proofs: if c is a commitment to $\mathbf{v}$ and a position needs to be updated resulting in a new commitment c′, an updatable VC must provide a method to update a valid claim of c into a valid claim of c′. The new opening should be computed by only knowing the portion of the vector that is changed and in time faster than recomputing the opening from scratch.

- *Homomorphic.* This property captures the ability to combine commitments and claims about openings. For example, given two commitments c and c′ to $\mathbf{v}$ and $\mathbf{v}'$, it should be possible to construct a commitment to $\mathbf{v} + \mathbf{v}'$ *without knowing* $\mathbf{v}, \mathbf{v}'$. Ideally, claims about openings should be homomorphic; for example, a claim about the i-th coordinate of c,c′ should be enough to construct a claim about the i-th opening of the commitment of $\mathbf{v} + \mathbf{v}'$.

- *Maintainability.* This property allows amortizing the proving costs in systems where committed values have a long life span and evolve over time. This is achieved by means of dedicating memory to reduce the computation time needed to open proofs. Concretely, the property requires that (1) one can efficiently store some values to reduce the cost of computing any individual openings (2) after updating a single position of the committed vector, it should be possible to update all proofs in time sublinear in the size of the vector (less than computing a single proof from scratch in some cases).

Our Contributions

In this chapter, we consider *combinatorial techniques* to construct vector commitment schemes. Specifically, we consider techniques similar to the ones used by Merkle trees, i.e. arranging the committed vector in a tree structure and commit "from leafs to root". We differentiate from Merkle trees in that we apply these techniques in the (plain and pairing) group setting which enjoys an algebraic structure, which is crucial for homomorphic properties. We only consider simple vector commitments that only satisfy position binding, although in some cases generalizations for slightly richer settings are possible. Concretely, we present two independent tree-based constructions with different goals in mind.

Vector commitment in the discrete logarithm setting. We construct a vector commitment scheme[2] in the plain discrete logarithm setting. We do this by first constructing a generic vector commitment scheme based on algebraic commitments and proofs of membership in linear space, and then instantiating with the Pedersen commitment and the folding technique. The construction has a *transparent setup* and has homomorphic commitments, but the proofs are not homomorphic, since the construction relies on the Fiat-Shamir transform which breaks the algebraic properties. Security holds in the Random Oracle model under the Discrete Logarithm assumption. Furthermore, the construction allows for some interesting trade-offs which are a direct consequence of the combinatorial structure of the tree: it allows pre-computing all opening proofs in time quasi linear in the size of the vector (instead of quadratic). Furthermore, it allows updating all proofs after a change in the committed vector in time equivalent to computing a single proof from scratch, i.e. linear in the size of the committed vector. We emphasize, however, that for performing this update, one needs to know the vector opening in its entirety; simply knowing the changed positions and the previous proofs is not enough.

Maintainable vector commitment with additional memory/time trade-offs. We present a *maintainable* vector commitment construction by exploiting the tensor structure of multivariate polynomials. This construction allows a stronger, more flexible form of maintainability: we achieve an arbitrary memory/time trade-off for openings, meaning that one can decide how much memory it wants to use to reduce the opening time accordingly. The construction is based on the Papamanthou et. al. polynomial commitment scheme [PST13] and generalizes Hyperproofs [SCP+22]. It improves on the latter in two different ways. First it utilizes low degree (instead of multilinear) polynomials which reduces the proof size by a constant factor. Second, it allows a very flexible memory/time trade-off: for vectors of size $k \cdot m$, one can pre-compute and store $O(m)$ elements and spend only $O(k)$ time to produce a proof. The proof size is independent of time parameter k. The stored parts of the proofs can be efficiently updated after a change in the committed vector.

5.1 Vector Commitment Definitions

In this section we recall definitions of vector commitments (VC), introduced by Catalano and Fiore [CF13], and subvector commitments (SVC), as presented in [BBF19; LM19].

Definition 32 (Vector Commitment)**.** A vector commitment for vectors from the message space $\mathcal{M}$ is a tuple of PPT algorithms $(\mathsf{KeyGen}, \mathsf{Com}, \mathsf{Open}, \mathsf{Verify})$ that work as follows:

$(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{VCS.KeyGen}(1^\kappa, \mathcal{M}, m)$**:** On input the security parameter κ, the message space $\mathcal{M}$ for the vectors and the maximum vector length m[3], it returns s *proving key* pk that

[2]Traditionally, the verification time of an opening is required to be sublinear in the size of the vector. This construction fails to achieve this property since it is based on the folding technique, which has a linear verifier. Nevertheless, this can be partially mitigated by amortizing the verification cost across many proofs using recent proof composition techniques.

[3]Some schemes are *unbounded*: they ignore m since they can commit to vectors of any length poly.

includes $\mathcal{M}$, and a *verification key* vk.

$(\mathsf{c}, \mathsf{aux}) \leftarrow \mathsf{VCS.Com}(\mathsf{pk}, \mathsf{vk})$**:** On input pk and a vector $\mathbf{v} = (v_1, v_2 \ldots, v_m) \in \mathcal{M}^m$, it returns a commitment c and auxiliary information aux.

$\pi_i \leftarrow \mathsf{VCS.Open}(\mathsf{pk}, \mathsf{aux}, i, v_i)$**:** On input pk, aux, an index $i \in [m]$, and a value v_i, it outputs a proof π_i that the value v_i is at position i.

$0/1 \leftarrow \mathsf{VCS.Verify}(\mathsf{vk}, \mathsf{c}, i, v, \pi_i)$**:** On input vk, c, $i \in [m]$, a value $v \in \mathcal{M}$ and π_i, it outputs 1 if the proof verifies and 0 otherwise.

that satisfies the following properties:

1. *Perfect Correctness.* A VC scheme is *perfectly correct* if, for all $\kappa \in \mathbb{N}$, any vector length $m = \mathsf{poly}(\kappa)$, any index $i \in [m]$, and any $\mathbf{v} \in \mathcal{M}^m$:

$$\Pr\left[\mathsf{VCS.Verify}(\mathsf{vk}, \mathsf{c}, i, v_i, \pi_i) = 1 \;\middle|\; \begin{array}{c}(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{VCS.KeyGen}(1^\kappa, \mathcal{M}, m) \\ (\mathsf{c}, \mathsf{aux}) \leftarrow \mathsf{VCS.Com}(\mathsf{pk}, \mathbf{v}) \\ \pi_i \leftarrow \mathsf{VCS.Open}(\mathsf{pk}, \mathsf{aux}, i, v_i)\end{array}\right] = 1$$

2. *Position Binding.* A VC scheme satisfies *position binding* if, for all PPT adversaries $\mathcal{A}$, for all $\kappa \in \mathbb{N}$, any vector length $m = \mathsf{poly}(\kappa)$:

$$\Pr\left[\begin{array}{c}\mathsf{VCS.Verify}(\mathsf{vk}, \mathsf{c}, i, v, \pi) = 1 \\ \wedge\, \mathsf{VCS.Verify}(\mathsf{vk}, \mathsf{c}, i, v', \pi') = 1 \\ \wedge\, v \neq v'\end{array} \;\middle|\; \begin{array}{c}(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{VCS.KeyGen}(1^\kappa, \mathcal{M}, m) \\ (\mathsf{c}, i, (v, \pi), (v', \pi')) \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{vk})\end{array}\right] \leq \mathsf{negl}(\kappa).$$

3. *Efficiency.* The size of the proof π should be *sublinear* in the size of the vector m.

One can also consider a weaker variant of position binding, where the adversary is additionally required to output a valid opening of the commitment c as well.

Next, we consider the definition of vector commitments that also support sub-vector openings.

Definition 33 (Vector Commitment with Subvector Opening)**.** A Vector Commitment with Sub-Vector Openings scheme is a VC scheme that opens subsets rather than simple positions. It consists on algorithms $(\mathsf{VCS.KeyGen}, \mathsf{VCS.Com}, \mathsf{VCS.Open}, \mathsf{VCS.Verify})$ that work as follows:

$(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{VCS.KeyGen}(1^\kappa, \mathcal{M}, m)$**:** On input the security parameter κ, the message space $\mathcal{M}$ for the vectors elements and the maximum vector length m and outputs a proving key pk and verification key vk.

$(\mathsf{c}, \mathsf{aux}) \leftarrow \mathsf{VCS.Com}(\mathsf{pk}, \mathbf{v})$**:** On input pk and a vector $\mathbf{v} = (v_1, v_2 \ldots, v_m) \in \mathcal{M}^m$, it outputs a commitment c and auxiliary information aux.

$\pi_S \leftarrow \mathsf{VCS.Open}(\mathsf{pk}, \mathsf{aux}, S, \mathbf{v}_S)$: On input pk, aux, a set of index $S \subset [m]$ and values $\mathbf{v}_S = \{v_i\}_{i \in S}$, it outputs a proof π_S that v_i is the value in position i, for all $i \in S$.

$0/1 \leftarrow \mathsf{VCS.Verify}(\mathsf{vk}, \mathsf{c}, S, \mathbf{y}, \pi_S)$: On input vk, c, a set S, a vector $\mathbf{y} = \{y_i\}_{i \in S}$ and π_I, it outputs 1 for accept or 0 for reject.

that satisfies the following properties:

1. *Perfect Correctness.* A VC scheme is *perfectly correct* if, for all $\kappa \in \mathbb{N}$, any vector length $m = \mathsf{poly}(\kappa)$, any set $S \subseteq [m]$, and any $\mathbf{v} \in \mathcal{M}^m$:

$$\Pr\left[\mathsf{VCS.Verify}(\mathsf{vk}, \mathsf{c}, S, \mathbf{v}_S, \pi_i) = 1 \;\middle|\; \begin{array}{c}(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{VCS.KeyGen}(1^\kappa, \mathcal{M}, m) \\ (\mathsf{c}, \mathsf{aux}) \leftarrow \mathsf{VCS.Com}(\mathsf{pk}, \mathbf{v}) \\ \pi_S \leftarrow \mathsf{VCS.Open}(\mathsf{pk}, \mathsf{aux}, S, \mathbf{v}_S)\end{array}\right] = 1$$

2. *Subvector Binding.* A VC scheme satisfies *position binding* if, for all PPT adversaries $\mathcal{A}$, for all $\kappa \in \mathbb{N}$, any vector length $m = \mathsf{poly}(\kappa)$:

$$\Pr\left[\begin{array}{c}\mathsf{VCS.Verify}(\mathsf{vk}, \mathsf{c}, S, \mathbf{v}, \pi) = 1 \\ \wedge\, \mathsf{VCS.Verify}(\mathsf{vk}, \mathsf{c}, S', \mathbf{v}', \pi') = 1 \\ \wedge\, \exists i \in S \cap S';\ \text{s.t. } v_i \neq v'_i\end{array} \;\middle|\; \begin{array}{c}(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{VCS.KeyGen}(1^\kappa, \mathcal{M}, m) \\ (\mathsf{c}, S, S', (v, \pi), (v', \pi')) \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{vk})\end{array}\right] \leq \mathsf{negl}(\kappa).$$

3. *Efficiency.* The size of the proof π_S should be *sublinear* in the size of the vector m and the size of the set S.

We next define some additional properties a vector commitment should satisfy. We consider these properties only in the case of simple Vector Commitments.

Definition 34 (Homomorphic Commitments and Openings.)**.** A vector commitment scheme is homomorphic if it supports an addition operation + on commitments such that for all κ, and $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{VCS.KeyGen}(1^\kappa, \mathcal{M}, m)$, if

$$(\mathsf{c}_1, \mathsf{aux}_1) \leftarrow \mathsf{VCS.Com}(\mathsf{pk}, \mathbf{v}_1), \qquad (\mathsf{c}_2, \mathsf{aux}_2) \leftarrow \mathsf{VCS.Com}(\mathsf{pk}, \mathbf{v}_2)$$

then $\tilde{\mathsf{c}} = (\alpha \mathsf{c}_1 + \beta \mathsf{c}_2)$ is a valid commitment to $\tilde{\mathbf{v}} = (\alpha \mathbf{v}_1 + \beta \mathbf{v}_2)$ for any $\alpha, \beta \in \mathcal{M}$. Additionally, it has homomorphic openings if it supports an addition operation + on proofs such that for all κ, $i \in [m]$, vectors $\mathbf{v}_1, \mathbf{v}_2 \in \mathcal{M}^m$ if

$$\pi_1 \leftarrow \mathsf{VCS.Open}(\mathsf{pk}, \mathsf{aux}_1, i, y_1), \qquad \pi_2 \leftarrow \mathsf{VCS.Open}(\mathsf{pk}, \mathsf{aux}_2, i, y_2)$$

then $\tilde{\pi} = (\alpha \pi_1 + \beta \pi_2)$ satisfies

$$\mathsf{VCS.Verify}\big(\mathsf{vk}, \tilde{\mathsf{c}} = (\alpha \mathsf{c}_1 + \beta \mathsf{c}_2), i, \tilde{y} = (\alpha y_1 + \beta y_2), \tilde{\pi}\big) = 1$$

We next consider updatability as an extra property of Vector Commitment schemes.

Definition 35 (Updatability)**.** A homomorhic vector commitment scheme is updatable if KeyGen algorithm additionally outputs an update key uk and there exists an algorithm Update such that

$\pi' \leftarrow \mathsf{VCS.Update}(\mathsf{uk}, \mathsf{j}, \delta, \mathsf{i}, \mathsf{y_i}, \pi_\mathsf{i})$**:** on input uk, j, δ, a position i, and a valid opening pair (y_i, π_i) for position i, it outputs a valid proof π'_i for the new commitment c′ defined as $\mathsf{c}' = \mathsf{c} + \mathsf{Com}(\delta \mathbf{e}_j)$ at position i.

Concretely, if y' is the new value of c′ at position i ($y' + \delta$ if $i = j$ and y otherwise), then

$$\mathsf{VCS.Verify}\left(\mathsf{vk}, \mathsf{c}' = (\mathsf{c} + \delta \mathbf{e}_j), i, y', \pi'\right) = 1$$

Note that updatability is implied by the homomorphic opening properties. Indeed, let c be a commitment and (y_i, π_i) a valid proof for the i-th coordinate. Now, consider the vector $\delta \mathbf{e}_j$, a commitment to it c^* and let (y_i^*, π_i^*) be a proof for its i-th coordinate. Note that $y_i^* = 0$ if $i \neq j$ and δ otherwise. Now, by the homomorphic openings property, $\tilde{\pi} = \pi_i + \pi_i^*$ is a valid proof for the i-th coordinate of the commitment $\tilde{\mathsf{c}} = \mathsf{c} + \mathsf{c}^*$. This is exactly the property that Updatability needs to satisfy. Note that we do not need to know the opening of c in its entirety; it is enough to know its opening and a proof for the i-th coordinate.

5.1.1 Algebraic Vector Commitments

We consider a specific family of vector commitments which we call algebraic. Essentially, a commitment scheme is algebraic if the commit algorithm is an inner product of elements of the commitment key with the committed vector. We consider such commitments in the group setting.

Definition 36 (Algebraic Vector Commitments)**.** Let $\mathsf{gk} \leftarrow \mathcal{G}(1^\kappa)$ be a description of a (plain or pairing group). A vector commitment scheme with message space $\mathbb{F}$ and commitment space $\mathbb{G}$ is algebraic if for committing to vectors of size m, the commitment key pk contains a vector $[\mathbf{r}] \in \mathbb{G}^m$ and the commit algorithm is defined as $\mathsf{VCS.Com}(\mathsf{pk}, \mathbf{v}_2) = [\mathbf{r}]^\top \mathbf{v}$.

All the constructions in this section are algebraic vector commitment scheme. Note that an algebraic vector commitment scheme is necessarily homomorphic. Indeed, if c_1 and c_2 are commitments to $\mathbf{v}_1, \mathbf{v}_2$ respectively, we have

$$\alpha \mathsf{c}_1 + \beta \mathsf{c}_2 = \alpha [\mathbf{r}]^\top \mathbf{v}_1 + \beta [\mathbf{r}]^\top \mathbf{v}_2 = [\mathbf{r}]^\top (\alpha \mathbf{v}_1 + \beta \mathbf{v}_2)$$

5.2 Vector Commitments in the Discrete Logarithm Setting

In this section we present a vector commitment scheme in the discrete logarithm setting. The construction allows for efficient *proof pre-computation*. To derive the construction we work as follows: first we present a generic vector commitment construction with *subvector openings* by using proof of knowledge of opening of an algebraic commitment. That is, we show that we know a representation of a group element w.r.t. a vector of group elements. While this construction is enough, it lacks efficient pre-computation of openings. We mitigate this by recursively applying this construction in a tree-like fashion to predefined subsets. The new tree structure adds redundancy to the proofs and allows to achieve the desirable pre-processing properties. Finally, we instantiate the PoK of opening using the folding technique, which yields a transparent vector commitment in the discrete logarithm setting (secure in the random oracle model).

5.2.1 Proof of Knowledge of Opening from the Folding Technique.

In this section we present proofs of knowledge of opening of a commitment schemes in the discrete logarithm setting, using the folding technique. Concretely, we define the **NP** language $\mathcal{L}_{\mathsf{gk},\mathbf{r}}$ parameterized by a group description gk and a commitment key $[\mathbf{r}] \in \mathbb{F}^m$, where m is the dimension of the committed vectors as

$$\mathcal{L}_{\mathsf{gk},\mathbf{r}} = \left\{[\mathsf{c}] \mid \exists \mathbf{x} \text{ s.t. } \mathsf{c} = \mathbf{r}^\top \mathbf{x}\right\}$$

and instantiate argument systems for this fixed language.

We do this by adapting the folding technique. We denote with $\mathbf{a}_0, \mathbf{a}_1$ the first and second half elements of a commitment $\mathbf{a}$. Intuitively, given an algebraic commitment $[\mathsf{c}] \in \mathbb{G}$, the prover convinces the verifier that it knows an opening $[\mathsf{c}] = [\mathbf{r}]^\top \mathbf{x}$ as follows:

- First, the prover gives commitments $[\mathsf{c}_1] = [\mathbf{r}_0]^\top \mathbf{x}_1$ and $[\mathsf{c}_0] = [\mathbf{r}_1]^\top \mathbf{x}_0$ and expects a random challenge $\chi \in \mathbb{F}$ from the verifier.

- Then it computes a new commitment key $[\mathbf{r}'] = [\mathbf{r}_0] + \chi^{-1}[\mathbf{r}_1]$ and a new witness $\mathbf{x}' = \mathbf{x}_0 + \chi \mathbf{x}_1$ and proceeds recursively in showing that a new commitment $[\mathsf{c}']$ opens to the new opening $\mathbf{x}'$ w.r.t. the new (halved) commitment key $[\mathbf{r}']$.

- When the witness is small enough -say constant- the prover simply presents the witness to the verifier.

From the verifier's perspective, the crucial point is that the new commitment key in each round is efficiently computable, and the new commitment can be computed from the commitments

Figure 5.1 Public coin protocol for proving knowledge of an opening of an algebraic commitment. We demonstrate the recursive step. When the witness is small, the prover simply sends it on the clear to the verifier.

$$\mathsf{x} = ([\mathbf{r}], [\mathsf{c}]), \mathsf{w} = \mathbf{a}$$

$\mathcal{P}: \mathsf{q}_i = (\mathsf{x}_i, \mathsf{w}_i)$		$\mathcal{V}: \mathsf{x}_i$
$[\mathsf{c}_0] = [\mathbf{r}_1]^\top \mathbf{a}_0, [\mathsf{c}_1] = [\mathbf{r}_0]^\top \mathbf{a}_1$	$\xrightarrow{[\mathsf{c}_0], [\mathsf{c}_1]}$	
	$\xleftarrow{\chi}$	$\chi \leftarrow \mathbb{F}$
$\mathbf{a}' = \mathbf{a}_0 + \chi \mathbf{a}_1$		
$[\mathsf{c}'] = [\mathsf{c}_1] + \chi[\mathsf{c}_2]$		$[\mathsf{c}'] = [\mathsf{c}_1] + \chi[\mathsf{c}_2]$
$[\mathbf{r}'] = [\mathbf{r}_0] + \chi^{-1}[\mathbf{r}_1]$		$[\mathbf{r}'] = [\mathbf{r}_0] + \chi^{-1}[\mathbf{r}_1]$
$\mathsf{x}' = ([\mathbf{r}'], [\mathsf{c}'])$		$\mathsf{x}' = ()[\mathbf{r}'], [\mathsf{c}'])$
$\mathsf{w}' = \mathbf{a}'$		

given to the verifier and the random challenge as follows:

$$\begin{aligned}
[\mathsf{c}_1]\chi + [\mathsf{c}] + [\mathsf{c}_0]\chi^{-1} &= [\mathbf{r}_0]^\top \mathbf{x}_1 \chi + [\mathbf{r}]^\top \mathbf{x} + [\mathbf{r}_0]^\top \mathbf{x}_1 \chi^{-1} \\
&= [\mathbf{r}_0]^\top \mathbf{x}_1 \chi + [\mathbf{r}_0]^\top \mathbf{x}_0 + [\mathbf{r}_1]^\top \mathbf{x}_1 + [\mathbf{r}_1]^\top \mathbf{x}_0 \chi^{-1} \\
&= [\mathbf{r}_0]^\top (\mathbf{x}_0 + \mathbf{x}_1 \chi) + [\mathbf{r}_1]^\top (\mathbf{x}_0 \chi^{-1} + \mathbf{x}_1) \\
&= [\mathbf{r}_0]^\top (\mathbf{x}_0 + \mathbf{x}_1 \chi) + [\mathbf{r}_1] \chi^{-1\top} (\mathbf{x}_0 + \mathbf{x}_1 \chi) \\
&= [\mathbf{r}_0 + \mathbf{r}_1 \chi^{-1}]^\top (\mathbf{x}_0 + \mathbf{x}_1 \chi)
\end{aligned}$$

As for soundness, intuitively, the prover must know openings of both $[\mathsf{c}_0]$ and $[\mathsf{c}_1]$ since it manages to open a random linear combination of them defined *after* the commitments are presented. Note that in each round (1) the witness size is halved and (2) a constant number of elements is communicated. This translates in a protocol with logarithmic rounds and communication. We present the protocol in Fig. 5.1. Note that this is a simpler variant of the folding technique used for inner product arguments. The proof of soundness is a direct adaptation of this case.

Remark (On Verification Efficiency)**.** The verification cost is linear in the dimension n. Nevertheless, most of the work done by the verifier is computing the new key $\mathbf{r}'$ which is fully determined by the initial key and the challenges and not the witness or the proof itself. This is an important observation that allows amortizing this cost across several proofs [BBB+18; BGH19; BCMS20].

5.2.2 Generic Construciton of Vector Commitments from PoK of Opening

In this section, we describe abstract techniques to construct tree-based vector commitments. Specifically, our construction only employs algebraic and combinatorial properties, making them very flexible in instantiating.

We start by showing an inherent connection between subvector openings and proofs of membership in linear spaces. Then we show how to utilize this technique and some simple combinatorial properties to derive tree-based vector commitment constructions with interesting properties. Finally, we instantiate the construction using the folding technique deriving a maintainable vector commitment scheme in the DLOG setting.

Non-membership Proofs

Consider an algebraic commitment scheme and a key $[\mathbf{r}]$. Now let $[\mathbf{c}]$ be a commitment and $S \subseteq \{1, \ldots, n\}$ a set of position we want to open. Denoting $\mathbf{x}$ the opening, we can write

$$[\mathbf{c}] = [\mathsf{c}]^\top \mathbf{x} = [\mathbf{r}_S]^\top \mathbf{x}_S + [\mathbf{r}_{\overline{S}}]^\top \mathbf{x}_{\overline{S}} = [\mathsf{c}_S] + [\mathsf{c}_{\overline{S}}]$$

where the subscripts S, $\overline{S}$ denote the vector components that belong and don't belong to S respectively.

A simple strategy for convincing the verifier about the validity of an opening of S positions $\mathbf{x}_S$ is to give to the verifier the part of the commitment not belonging to S, namely $[\mathsf{c}_{\overline{S}}]$ and a proof of knowledge of an opening for it w.r.t. to the corresponding key $\mathbf{r}_{\overline{S}}$. The verifier will then (1) assert the proof of opening and (2) reconstruct the part $[\mathsf{c}_S] = [\mathbf{r}_S]^\top \mathbf{x}_S$ using the claimed subvector opening $\mathbf{x}_S$ and assert that $[\mathsf{c}] = [\mathsf{c}_S] + [\mathsf{c}_{\overline{S}}]$.

This simple construction indeed works. Intuitively, because the proof of knowledge does not involve the parts of the key that are in S, the opening that the prover knows for $\overline{S}$ and the claimed opening for S are independent. Thus, opening the variables in S in two different ways would imply that the prover knows two openings for the commitment $[\mathsf{c}]$ itself; this cannot happen due to the binding property.

Essentially, what we describe can be thought as a *non-membership* proof: we show that the commitment opening $\mathsf{c}_{\overline{S}}$ *is not* constructed as a linear combination of the S part of the commitment key; in some sense, it does not belong in the linear space generated by $[\mathbf{r}_S]$ from the perspective of the prover[4]. We call such proofs *non-membership proofs*.

How do we show that something does not belong in some space? It is enough to show that it belongs in a different space! This only works because if we knew two ways of representing $[\mathsf{c}]$ w.r.t. distinct elements, we would break a FindRep assumption.

[4]Of course the commitment trivially lies in the said linear space. The prover shows that it does not know a way to represent it as such.

Figure 5.2 Vector commitment construction using non-membership proofs. $\mathcal{D}_{n,1}$ is a matrix distribution where the FindRep assumption holds and NARK is an argument system for proving membership in linear spaces.

VCS.KeyGen$(1^\kappa, m)$**:**

Sample $[\mathbf{r}] \leftarrow \mathcal{D}_{m,1}$

For each $S \subseteq \{1, \ldots, m\}$ compute $(\mathsf{srs}_S, \tau_S) \leftarrow \mathsf{NARK.Setup}_{\mathcal{L}_{\mathsf{gk},[\mathbf{r}_S]}}(\mathsf{gk})$

Output $\mathsf{pk} = ([\mathbf{r}], \{\mathsf{srs}_S\}_S)$ and $\mathsf{vk} = ([\mathbf{r}], \{\mathsf{srs}_S\}_S)$

VCS.Com(ck, $\mathbf{x}$)**:**

Output $[\mathsf{c}] = [\mathbf{r}]^\top \mathbf{x}$, $\mathsf{aux} = \mathbf{x}$

VCS.Open(pk, aux, $S, \mathbf{y}$)**:**

Set $[\mathsf{c}_{\overline{S}}] = [\mathsf{c}] - [\mathbf{r}_S]^\top \mathbf{x}_S$

Output $\pi = \mathsf{NARK.Prove}(\mathsf{srs}_{\overline{S}}, [\mathsf{c}_{\overline{S}}], \mathbf{x}_{\overline{S}})$

VCS.Verify(vk, $[\mathsf{c}], S, \mathbf{y}, \pi$)**:**

Set $[\mathsf{c}_{\overline{S}}] = [\mathsf{c}] - [\mathbf{r}_S]^\top \mathbf{y}$

Output 1 if $\mathsf{NARK.Verify}(\mathsf{srs}_{\overline{S}}, [c_{\overline{S}}], \pi)$ and 0 otherwise

So, the construction for opening commitment $[\mathsf{c}]$ at $\mathbf{x}_S$ is essentially showing that $[\mathsf{c}_{\overline{S}}] = [\mathsf{c}] - [\mathbf{r}_{\overline{S}}]$ belongs in the linear subspace generated by $[\mathbf{r}_{\overline{S}}]$. In other words, proving knowledge of opening of $[\mathsf{c}_{\overline{S}}]$. Instantiating with any algebraic commitment and proof of knowledge is enough.

We present the construction more formally in Fig 5.2. Note that key generation might be inefficient. Specifically, one would need one membership in linear space srs for each $S \subseteq [m]$. Even if we restrict the subsets to ones that have cardinality 1, this would still be quadratic in the dimension of the vector. While this can be a big efficiency barrier in general, this is not the case in the instantiation of the folding technique as we will see next. The reason is that the different srses share elements, resulting in a single srs of size linear in the vector dimension. Furthermore, regarding the more general setting, the intention of this construction is to be used as a black box in a tree based commitment scheme. In this intended use, we do not care about the ability to open all (exponential in size) sets, but rather we focus in opening some well chosen ones iteratively, thus countering the aforementioned inefficiency.

We next present a theorem showing that the construction is position binding and the efficiency properties of it. We explicitly do not mention commitment key size; we consider this only in a per case base.

Theorem 30. *Let $\{\mathcal{D}_{m,1}\}_{m\in\mathbb{N}}$ be any family of matrix distributions where the* FindRep *assumption is hard and* NARK *be a non-interactive argument of knowledge for the membership in linear spaces family of languages. Then, for any $m \in \mathbb{N}$ construction of Fig. 5.2 is a vector commitment scheme*

with subvector opening that satisfies position binding. Furthermore, if $t_{\mathcal{P}}(k), \pi(k), t_{\mathcal{V}}(k)$ are the proving time, proof size and verification time of NARK *for instances of size k, then the time of opening a subset S is $t_{\mathcal{P}}(m - |S|)$, proof of opening is of size $\pi(m - |S|)$ and verification time is $t_{\mathcal{V}}(m - |S|)$.*

Proof. We only focus on the position binding property since completeness and efficiency follows from the description of the construction in a straightforward manner.

Let $\mathcal{A}$ be an adversary that outputs two valid openings for a commitment $[\mathsf{c}]$, namely $([\mathsf{c}], S_0, S_1, \mathbf{y}_0, \mathbf{y}_1, \pi_0, \pi_1)$ such that $\mathsf{Verify}(\mathsf{vk}, [\mathsf{c}], S_b, \mathbf{y}_b, \pi_b) = 1$ for both $b \in \{0, 1\}$. Denote by S the intersection of S_0, S_1. By knowledge soundness of NARK, we can extract two openings

- $\mathbf{x}'_0$ for $[\mathsf{c}] - [\mathbf{r}_{S_0}]^\top \mathbf{y}_0$ w.r.t. key $[\mathbf{r}_{\overline{S}_0}]$,
- $\mathbf{x}'_1$ for $[\mathsf{c}] - [\mathbf{r}_{S_1}]^\top \mathbf{y}_1$ w.r.t. key $[\mathbf{r}_{\overline{S}_1}]$.

This means that we can write $[c]$ as

$$[\mathbf{r}_{S_0}]^\top \mathbf{y}_0 + [\mathbf{r}_{\overline{S}_0}]^\top \mathbf{y}'_0 = [\mathsf{c}] = [\mathbf{r}_{S_1}]^\top \mathbf{y}_0 + [\mathbf{r}_{\overline{S}_1}]^\top \mathbf{y}'_1$$

or equivalently

$$[\mathbf{r}]^\top (\mathbf{y}_0, \mathbf{y}'_0) = [\mathsf{c}] = [\mathbf{r}]^\top (\mathbf{y}_1, \mathbf{y}'_1)$$

Assuming that $\mathcal{A}$ wins, the claimed openings $\mathbf{y}_0, \mathbf{y}_1$ should differ in at least one position in the intersection S which means that $(\mathbf{y}_0, \mathbf{y}'_0) \neq (\mathbf{y}_1, \mathbf{y}'_1)$ which happens only with negligible probability since otherwise we break the $\mathcal{D}_{n,1}$-FindRep assumption. □

Instantiation. We only consider an instantiation using the folding technique. The srs for opening subset S is simply $\mathbf{r}_S$; that is, no additional elements are needed apart from the commitment key as well. In fact, there is no need for a key generation algorithm at all, simply sampling a commitment key from a matrix distribution is enough! By directly applying the above theorem, we get a publicly verifiable construction with linear prover and verifier and a logarithmic proof size. We need to rely on the random oracle and the Fiat-Shamir transform for position binding. Using the uniform distribution over $\mathbb{F}^m$ results in a *transparent* VC scheme. The disadvantage of using the folding technique is the verification time. We only consider the publicly verifiable setting, but if one is willing to settle for either the designated verifier setting or the pairing group setting and a trusted setup, we can mitigate the linear verification time using the techniques of Chapter(**alex**:Fill)

Recursive non-membership proofs

In this section we utilize the non-membership approach combined with some combinatorial techniques to improve on efficiency and achieve additional properties and trade-offs. Specif-

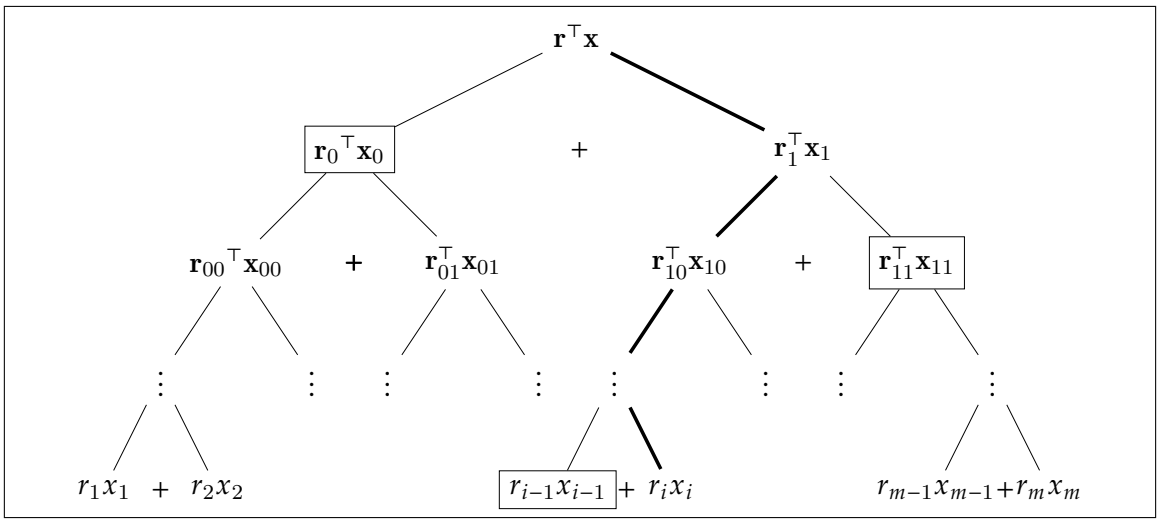


Figure 5.3: Demonstration of the process of opening coordinate i. Bold edges denote the path the prover follows and rectangles the commitments and proofs of openings of the siblings the prover has to serve as a proof. In the middle nodes we denote with $\mathbf{r}_{b_1\ldots b_j}$ the commitment key that contains the elements r_ℓ of ℓ whose binary representation is prefixed by $(b_1,\ldots,b_j)$ (resp. for the openings)

ically, the approach allows us to achieve *proof pre-computation*, namely the ability to pre-compute all individual position proofs of a committed vector in non-trivial time. Here, non-trivial means asymptotically less that computing naively all individual proofs. In our instantiation using the folding technique, the pre-computing time is quasi linear (instead of quadratic).

Our instantiation does not support updates. This is because it does not have *homomorphic proofs* due the inherent reliance on the Fiat-Shamir transform which breaks homomorphicity. Nevertheless, assuming that we know the whole opening (and not just the modified positions), we can *update all proofs* in time equivalent to computing from scratch a single proof.

Recursive non-membership proof construction. Consider a prover, wishing to convince a verifier about the value of an opening of the i-th coordinate of some committed vector. We showcase the high level idea of using recursive non-membership proofs next.

Because the commitment is algebraic, we can split the commitment in two independent spaces, each containing half of the variables. That is, if the commitment key $\mathbf{r} = (\mathbf{r}_0, \mathbf{r}_1)$ and the committed value is $\mathbf{x} = (\mathbf{x}_0, \mathbf{x}_1)$, where each of $\mathbf{r}_b$, $\mathbf{x}$ contain $n/2$ elements, then we can write

$$[\mathsf{c}] = [\mathbf{r}]^\top\mathbf{x} = [\mathbf{r}_0]^\top\mathbf{x}_0 + [\mathbf{r}_1]^\top\mathbf{x}_1 = [\mathsf{c}_0] + [\mathsf{c}_1]$$

Now, if the coordinate i we want to open belongs to -say- the first half variables, namely $\mathbf{x}_0$, it would be enough for the prover to open the commitment $[\mathsf{c}_0]$ w.r.t. the key $[\mathbf{r}_0]$. Obviously, this is both redundant -since we only care about one variable- and inefficient; nevertheless, we can overcome both issues by having the prover give a proof of knowledge instead. Now it is enough to solve a smaller subproblem, namely, opening the i-th coordinate of $[\mathsf{c}_0]$ w.r.t. the

key $[\mathbf{r}_0]$. And, of course, the way to do it is recursion!

A bit more concretely, in order to convince the verifier that for some $i \leq m/2$ the opening of $[\mathsf{c}]$ in the i-th coordinate is x_i, the prover proceeds as follows:

1. It writes the commitment as $[\mathsf{c}] = [\mathsf{c}_0] + [\mathsf{c}_1]$ and sends $[\mathsf{c}_0], [\mathsf{c}_1]$[5]

2. It sends a *non-membership proof* of $[\mathsf{c}_0]$ in the space spanned by $[\mathbf{r}_1]$

3. It recursively shows that the i-th opening of $[\mathsf{c}_0]$ w.r.t. the first half of the key $[\mathbf{r}_0]$

From the verifier's perspective, it is enough to check the proof of non-membership of $[\mathsf{c}_0]$ as well as that the "splitting" of the commitment is correct, namely that $[\mathsf{c}] = [\mathsf{c}_0] + [\mathsf{c}_1]$. At the base of the recursion, we simply have a claim $[\mathsf{c}_i] = [r_i]x_i$ which the verifier can assert on its own. We present a pictorial figure of the construction in fig. 5.3.

We formally present the construction in Fig. 5.4. We use as a black box a non-interactive argument NARK = (Setup, Prove, Verify) for membership in linear spaces and an algebraic commitment scheme. The commitment key is parameterized by a family of matrix distributions $\{\mathcal{D}_{m,1}\}_{m\in\mathbb{N}}$.

Note that we can modify easily the construction for the case where the prover wants to convince the verifier that it knows that opening of position i instead of communicating it: in this case the prover simply sends a PoK of opening of the leaf commitment $[r_i]x_i$ instead of the opening itself. In the final step the verifier simply asserts the validity of the proof.

Theorem 31. *Let* NARK *be a non-interactive argument of knowledge. Then construction of Fig. 5.4 is an algebraic vector commitment scheme that satisfies position binding under the* $\mathcal{D}_{1,m}$-FindRep *assumption.*

Proof. Completeness follows by inspection. We present position binding next. We only consider the case where the adversary outputs a subset of indices of a vector of cardinality one.

Assume that an adversary $\mathcal{A}$ outputs a commitment and two openings $[\mathsf{c}], \{i\}, x, x^*, \pi, \pi^*$ such that $\mathsf{Verify}(\mathsf{vk}, [\mathsf{c}], \{i\}, x, \pi) = \mathsf{Verify}(\mathsf{vk}, [\mathsf{c}], \{i\}, x^*, \pi^*) = 1$ and $x \neq x^*$. First, we focus on one of the two proofs. Let $(b_v, \ldots, b_1)$ be the binary representation of the opening

[5]The prover needs to send one of these since $[\mathsf{c}_{1-b}] = [\mathsf{c}] - [\mathsf{c}_b]$. In what follows, we use the convention that the prover sends the part of the key not containing i as a parallelism to the Merkle construction.

Figure 5.4 Vector commitment construction using non-membership proofs recursively. We assume that the dimension of the committed vectors is $m = 2^\nu$.

$\mathsf{KeyGen}(1^\kappa, m)$**:**

$\mathbf{r} \leftarrow \mathcal{D}_{1,m}$

For all $(b_\nu, \ldots, b_\ell)$ with $b_j \in \{0,1\}, \ell \in \{\nu, \ldots, 1\}$ compute an srs-trapdoor pair for proving knowledge of an opening w.r.t. the commitment key $[\mathbf{r}_{b_\nu \ldots b_\ell}]_1$.

$(\mathsf{srs}_{b_\nu \ldots b_\ell}) \leftarrow \mathsf{NARK.Setup}(\mathsf{par}, [\mathbf{r}_{b_\nu \ldots b_\ell}])$

$\mathsf{pk} := \left([\mathbf{r}], \left\{\mathsf{srs}_{b_\nu \ldots b_\ell}\right\}_{b_j \in \{0,1\}, \ell \in \{\nu,\ldots,1\}}\right)$, $\mathsf{vk} := \left([\mathbf{r}], \left\{\mathsf{srs}_{b_\nu \ldots b_\ell}\right\}_{b_j \in \{0,1\}, \ell \in \{\nu,\ldots,1\}}\right)$

Output $(\mathsf{ck}, \mathsf{vk})$

$\mathsf{Com}([\mathbf{r}], \mathbf{x})$**:**

Output $[\mathsf{c}] := [\mathbf{r}]^\top \mathbf{x}$, $\mathsf{aux} = \mathbf{x}$

$\mathsf{Open}(\mathsf{pk}, \mathsf{aux}, i = (b_\nu, \ldots, b_1), y)$**:** If $\nu = 1$ output nothing, otherwise

$[\mathsf{c}_{\overline{b}_\nu}] = [\mathbf{r}_{\overline{b}_\nu}]^\top \mathbf{x}_{\overline{b}_\nu}$

$\pi_{\overline{b}_\nu} \leftarrow \mathsf{NARK.Prove}(\mathsf{srs}_{\overline{b}_\nu}, [\mathsf{c}_{\overline{b}_\nu}], x_{\overline{b}_\nu})$

Output $[\mathsf{c}_{\overline{b}_\nu}], \pi_{\overline{b}_\nu}$

Recursively run $\mathsf{Open}(\mathsf{ck}, i = (b_{\nu-1}, \ldots, b_1), [\mathsf{c}] - [\mathsf{c}_{\overline{b}_\nu}], \mathbf{x}_{b_\nu})$

$\mathsf{Verify}(\mathsf{vk}, [\mathsf{c}], i = (b_\nu, \ldots, b_2), y, \pi)$**:**

Parse $\pi = \left([\mathsf{c}]_{\overline{b}_\nu}, \pi_{\overline{b}_\nu}, \ldots, [\mathsf{c}]_{b_\nu \ldots \overline{b}_1}, \pi_{b_\nu \ldots \overline{b}_1}\right)$ and $[\mathsf{c}_{b_\nu}] = [\mathsf{c}] - [\mathsf{c}_{\overline{b}_\nu}]$.

For each $\ell \in \{\nu, \ldots 2\}$

– If $\mathsf{NARK.Verify}(\mathsf{srs}_{b_\nu \ldots \overline{b}_\ell}, [\mathsf{c}_{b_\nu \ldots \overline{b}_\ell}], \pi_{b_\nu \ldots \overline{b}_\ell}) = 0$ output 0.

– Otherwise set $[\mathsf{c}_{b_\nu \ldots b_{\ell-1}}] = [\mathsf{c}_{b_\nu \ldots b_\ell}] - [\mathsf{c}_{b_\nu \ldots \overline{b}_{\ell-1}}]$

Finally, output 1 iff $[\mathsf{c}_{b_\nu,\ldots,b_1}] = [r_{b_\nu,\ldots,b_1}]^\top \, x_{b_\nu,\ldots,b_1}$

position i. Since the verifier accepts we have that

$$
\begin{aligned}
\mathsf{c} &= \mathsf{c}_{b_\nu} + \mathsf{c}_{\overline{b}_\nu} \\
\mathsf{c}_{b_\nu} &= \mathsf{c}_{b_\nu b_{\nu-1}} + \mathsf{c}_{b_\nu \overline{b}_{\nu-1}} \\
&\vdots \\
\mathsf{c}_{b_\nu \ldots b_\ell} &= \mathsf{c}_{b_\nu \ldots b_{\ell-1}} + \mathsf{c}_{b_\nu \ldots \overline{b}_{\ell-1}} \\
&\vdots \\
\mathsf{c}_{b_\nu \ldots b_2} &= \mathsf{c}_{b_\nu \ldots b_1} + \mathsf{c}_{b_\nu \ldots \overline{b}_1} \\
\mathsf{c}_{b_\nu \ldots b_1} &= r_{b_\nu \ldots b_1} x
\end{aligned}
$$

Note that we can extract all the openings of the commitments in the right hand side of the equations. Indeed, by knowledge soundness of NARK we can extract an opening $\mathbf{x}_{b_\nu \ldots \overline{b}_{\ell-1}}$ such that $\mathsf{c}_{b_\nu \ldots \overline{b}_{\ell-1}} = \mathbf{r}_{b_\nu \ldots \overline{b}_{\ell-1}}{}^\top \mathbf{x}_{b_\nu \ldots \overline{b}_{\ell-1}}$ and we can assume inductively that we can extract an opening $\mathbf{x}_{b_\nu \ldots b_{\ell-1}}$ such that $\mathsf{c}_{b_\nu \ldots b_{\ell-1}} = \mathbf{r}_{b_\nu \ldots b_{\ell-1}}{}^\top \mathbf{x}_{b_\nu \ldots b_{\ell-1}}$. The concatenation of these openings is an opening for $\mathsf{c}_{b_\nu \ldots b_\ell}$ w.r.t. the key $\mathbf{r}_{b_\nu \ldots b_\ell}$. For the base of the induction we use the fact that the claimed opening x satisfies $x = x_{b_\nu \ldots b_1}$ and $\mathsf{c}_{b_\nu \ldots b_1} = r_{b_\nu \ldots b_1} x_{b_\nu \ldots b_1}$. An identical argument holds for the values of the proof π^* for the opening x^*.

Now, from the above equations we see that (1) $\mathsf{c}_{b_\nu \ldots b_1} \neq \mathsf{c}^*_{b_\nu \ldots b_1}$ because $x \neq x^*$ and (2) $\mathsf{c} = \mathsf{c}^*$. This means that there should exist a minimal k with $2 \leq k \leq \nu + 1$ s.t.

$$\mathsf{c}_{b_\nu \ldots b_k} = \mathsf{c}^*_{b_\nu \ldots b_k} \quad \text{and} \quad \mathsf{c}_{b_\nu \ldots b_{k-1}} \neq \mathsf{c}^*_{b_\nu \ldots b_{k-1}}$$

where we set $\mathsf{c}_{b_\nu \ldots b_{\nu+1}} = \mathsf{c}$ and $\mathsf{c}^*_{b_\nu \ldots b_{\nu+1}} = \mathsf{c}^*$.

We next consider the openings for this minimal k. Let $\mathbf{x}_k, \mathbf{x}^*_k, \mathbf{x}_{k-1}, \mathbf{x}^*_{k-1}$ be such openings. Then the following relations hold:

$$\mathsf{c}_{b_\nu \ldots b_k} = \mathbf{r}_{b_\nu \ldots b_k}{}^\top \mathbf{x}_k = \mathbf{r}^\top_{b_\nu \ldots b_k} \mathbf{x}^*_k = \mathsf{c}^*_{b_\nu \ldots b_k}$$

$$\mathsf{c}_{b_\nu \ldots b_{k-1}} = \mathbf{r}_{b_\nu \ldots b_{k-1}}{}^\top \mathbf{x}_{k-1} \neq \mathbf{r}_{b_\nu \ldots b_k}{}^\top \mathbf{x}^*_{k-1} = \mathsf{c}^*_{b_\nu \ldots b_{k-1}}$$

$$\mathsf{c}_{b_\nu \ldots \overline{b}_{k-1}} = \mathbf{r}_{b_\nu \ldots \overline{b}_{k-1}}{}^\top \mathbf{x}_{\overline{k-1}}$$

$$\mathsf{c}^*_{b_\nu \ldots \overline{b}_{k-1}} = \mathbf{r}_{b_\nu \ldots \overline{b}_{k-1}}{}^\top \mathbf{x}^*_{\overline{k-1}}$$

We consider different cases for the extracted openings.

- If $\mathbf{x}_k \neq \mathbf{x}^*_k$, we break the $\mathcal{D}_{n,1}$-KerMDH assumption for $\mathbf{r}$ since $\mathbf{r}_{b_\nu \ldots b_k}{}^\top \mathbf{x}_k = \mathbf{r}_{b_\nu \ldots b_k}{}^\top \mathbf{x}^*_k$ (we can use 0 for the elements in the positions not in $\mathbf{r}_{b_\nu \ldots b_k}$).
- If $\mathbf{x}_k \neq \begin{pmatrix} \mathbf{x}_{k-1} \\ \mathbf{x}_{\overline{k-1}} \end{pmatrix}$, since $\mathsf{c}_{b_\nu \ldots b_k} = \mathsf{c}_{b_\nu \ldots b_{k-1}} + \mathsf{c}_{b_\nu \ldots \overline{b}_{k-1}}$ we get

  $$\mathbf{r}_{b_\nu \ldots b_k}{}^\top \mathbf{x}_k = \mathbf{r}_{b_\nu \ldots b_{k+1}}{}^\top \mathbf{x}_{k+1} + \mathbf{r}_{b_\nu \ldots \overline{b}_{k+1}}{}^\top \mathbf{x}_{\overline{k+1}} = \mathbf{r}_{b_\nu \ldots b_k}{}^\top \begin{pmatrix} \mathbf{x}_{\overline{k-1}} \\ \mathbf{x}_{\overline{k-1}} \end{pmatrix}$$

  and we again break the $\mathcal{D}_{n,1}$-KerMDH assumption using the values $\mathbf{x}_k, \begin{pmatrix} \mathbf{x}_{k-1} \\ \mathbf{x}_{\overline{k-1}} \end{pmatrix}$ (again filled with zeros in the rest of positions).
- A symmetric argument works for the case $\mathbf{x}^*_k \neq \begin{pmatrix} \mathbf{x}^*_{k-1} \\ \mathbf{x}^*_{\overline{k-1}} \end{pmatrix}$

Excluding the above cases it must be the case that $\mathbf{x}_{k-1} = \mathbf{x}^*_{k-1}$ which contradicts the minimality condition $\mathsf{c}_{b_\nu \ldots b_{k-1}} \neq \mathsf{c}^*_{b_\nu \ldots b_{k-1}}$.

□

Folding Technique Instantiation. In this section we consider an instantiation derived from the constructions of Fig. 5.4 using the folding technique NARK from Fig. 5.1. Importantly, the construction allows to compute the proofs for all possible coordinates in time less than what is needed to compute each proof individually. This stems from the combinatorial structure of the tree.

We also emphasize that if one is willing to use groups equipped with bilinear maps, the construction of chapter (**alex**:FILL) can be used to improve verification efficiency. This comes, however, with the need for a structure and therefore trusted setup. We summarize the efficiency properties of the instantiations next.

Corollary 7. *Construction of fig. 5.4 where we instantiate the* NARK *using the construction of fig. 5.1 is a vector commitment scheme. It satisfies the following efficiency properties for vectors of dimension* $n = 2^v$*:*

- *The size of the* srs *is* $|\mathsf{srs}| = n$.
- *The time to compute a single proof is* $O_\kappa(n)$.
- *The time to pre-compute all proofs is* $O_\kappa(n \log n)$.
- *The proof size is* $O_\kappa(\log^2 n)$.
- *The time to verify a proof is* $O_\kappa(n)$.

The proof of the corollary is a consequence of Thm. 31 and Thm. 30. Efficiency follows by simple counting arguments. The construction is not maintainable due to the lack of homomorphic properties of the underlying proof system: while one can easily modify the commitment itself, the previous proofs are completely useless after the update since they inherently rely on the statement due to the use of the Fiat-Shamir heuristic; thus, one needs to recompute the proofs from scratch. To do so, it is necessary to know and use the whole opening of the committed vector itself.

Nevertheless, if all proofs are pre-computed, one simply needs time linear in n to *update all proofs* after a single update (than recomputing the proofs from scratch). This holds because only the path from the root of the tree to the changed leaf is affected.

5.3 Memory-Time Tradeoffs for Vector Commitments

One of the key points of vector commitment schemes that allow aggregation of proofs is the ability to pre-compute and store individual openings and later use them to create subvector openings without incurring linear amount of computations each time.

A plethora of vector commitment schemes have constant proof size. While this is a very attractive and important property in practice, schemes with this property share a common limitation regarding pre-processing and updating proofs. A proof of opening of one position involves elements in *all* other positions of the vector. That is, the polynomials committed to create the proof have coefficients that involve all the values of the committed vector $\mathbf{v} \in \mathbb{F}^m$. As a consequence, prover work is linear in the size of $\mathbf{v}$ (as it has to evaluate polynomials of degree m) and after updating one position of $\mathbf{v}$, all m proofs of opening are changed.

This makes *pre-computing* and *maintaining* proofs impractical in the case of applications where commitments evolve over time. Furthermore, no useful memory-time trade-offs can be considered for this case.

In this section, we present a solution for more efficient openings for individual positions that is maintainable. The intuition is the following: we divide the vector $\mathbf{v}$ in small chunks $\{\mathbf{v}_j\} \in \mathbb{F}^k$. We then arrange these chunks in a tree as follows: each chunk corresponds to a leaf of the tree and each node is a succinct representation of its children. The root of the tree is the committed value. An opening proof only involves the elements in the path of the root to the leaf containing the position to be opened. That is, if we want to open value a in position i of $\mathbf{v} \in \mathbb{F}^{k \cdot m'}$, we prove that (1) c_j is the leaf that contains the commitment to the j chunk containing i and (2) c_j opens to a in the position corresponding to i. The former part can be pre-computed and efficiently maintained, occupying storage linear in m', while the latter involves operations that are linear in k.

This results in a construction with the following memory/time trade-off: for any $k, m' \in \mathbb{N}$ with $m = k \cdot m'$, any opening can be computed in time *independent of* m' after pre-computing and storing $O_\kappa(m')$ values (independent of k). Furthermore, a relaxed *maintainability* notion is satisfied: all stored values can be pre-computed efficiently (in quasi linear time in m) and updated in $\log m'$ time.

Our starting point is the PST polynomial commitment [BMM+21]. We first show how to directly use its low degree variant (instead of the multilinear one used in [SCP+22]) to reduce proof size by a constant factor. Second, we modify this construction by composing it with any other algebraic[6] vector commitment scheme to achieve maximum flexibility in time/memory trade-offs.

We next recall the PST polynomial commitment and describe the core techniques to use it as a maintainable vector commitment.

[6]We call algebraic, any (vector) commitment scheme where the commit algorithm works by simply setting $[c]_1 = [\mathbf{r}]_1^\top \mathbf{v}$, where $[\mathbf{r}]_1$ is the commitment key and $\mathbf{v}$ is the message to be committed.

5.3.1 PST Polynomial Commitment

The PST polynomial commitment allows to commit to ν-variate polynomials of individual degrees less than ℓ. The core idea of the construction lies in the fact that for every $p(\mathbf{X}) \in \mathbb{F}[X_\nu, \ldots, X_1]$ and $\mathbf{x} = (x_\nu, \ldots, x_1) \in \mathbb{F}^\nu$, $p(\mathbf{x}) = y$ if and only if there exist polynomials $H_\nu(\mathbf{X}), \ldots, H_1(\mathbf{X})$ such that

$$p(\mathbf{X}) - y = \sum_{j=1}^{\nu} H_j(\mathbf{X}) \cdot (X_j - x_j)$$

where the proof polynomials $H_j(\mathbf{X})$ are efficiently computable.

Using standard techniques to encode monomials in a cryptographically secure bilinear group (encode setting $\mathbf{X} = \boldsymbol{\tau}$ and publishing all the monomials $[\tau_\nu^{d_\nu} \cdots \tau_1^{d_1}]_1$ and $[\boldsymbol{\tau}]_2$) results in a polynomial commitment with proof of size roughly ν group elements

Srinivasan et. al. [SCP+22] observe that computing all polynomial evaluations and proofs for a committed ν-variate multilinear polynomial in the hypercube $\{0,1\}^\nu$ can be done in quasi-linear time in the dimension of the vector, instead of the trivial quadratic time. By encoding a vector as the corresponding interpolating polynomial in $\{0,1\}^\nu$, we get a vector commitment with quasi-linear time for pre-computing all proofs. Furthermore, the homomorphic properties along with the tensor structure of multivariate polynomials allow to efficiently (in logarithmic time) update all proofs after a position update. Thus, the resulting construction is a *maintainable* vector commitment scheme.

We extend these techniques to construct a multi-variate vector commitment scheme with the same properties while reducing proof size. Specifically, we observe that evaluating all openings in any set of the form Σ^ν for Σ with small size (i.e. constant) has lower amortized cost than computing the evaluations individually. Using Σ with $|\Sigma| > 2$ -or equivalently using a low degree instead of a multilinear encoding- results in smaller proof size. Concretely, the proof size depends on the dimension of the hypercube. Setting $\ell = O(1)$ to avoid a blowup in the prover's computation results in proof size roughly $\log_\ell m$ instead of $\log_2 m$, reducing the proof size by a constant factor.

We next describe the high level idea of the construction. First, fix an alphabet $\Sigma \subseteq \mathbb{F}$ of size ℓ and consider the hypercube Σ^ν. Assume (w.l.o.g.) $\Sigma = \{0, \ldots, \ell - 1\}$ so that we can encode indices of vectors in ℓ-ary.

Now, we can encode a vector $\mathbf{v} \in \mathbb{F}^{\ell^\nu}$ by considering the (unique) low degree interpolating polynomial $p(\mathbf{X})$ of $\mathbf{v}$, that is, the ν-variate polynomial of individual degree less than ℓ such that for all $\boldsymbol{\sigma} \in \Sigma^\nu$, $p(\boldsymbol{\sigma}) = v_\sigma$. This corresponds to position i with ℓ-ary representation $(\boldsymbol{\sigma})_\ell$. Computing all opening proofs corresponds to evaluating and proving evaluations of $p(\mathbf{X})$ in the hypercube Σ^ν. To compute these evaluations in quasi-linear (instead of quadratic) time we rely on the following lemma which is implicit in the computation of the $H_j(\mathbf{X})$ polynomials of PST.

Lemma 5. *Let $\Sigma \subseteq \mathbb{F}$ be a subset of $\mathbb{F}$. Also, let $p(X_\nu, \ldots, X_1) \in \mathbb{F}[X_\nu, \ldots, X_1]$ be a polynomial in ν variables and $p_\sigma(X_{\nu-1}, \ldots, X_1) \in \mathbb{F}[X_{\nu-1}, \ldots, X_1]$ be a polynomial in $\nu - 1$ variables. Then, for all $\sigma \in \Sigma$, $p(\sigma, X_{\nu-1} \ldots, X_1) = p_\sigma(X_{\nu-1} \ldots, X_1)$ iff there exist a polynomial $H(X_\nu, \ldots, X_1)$ such that*

$$p(X_\nu, \ldots, X_1) - p_\sigma(X_{\nu-1}, \ldots, X_1) = H(X_\nu, \ldots, X_1)(X_\nu - \sigma) \tag{5.1}$$

To open the polynomial at $\boldsymbol{\sigma} = (\sigma_\nu, \ldots, \sigma_1) \in \Sigma^\nu$, the prover can compute the polynomials $p(\sigma_\nu, X_{\nu-1}, \ldots, X_1), \ldots, p(\sigma_\nu, \ldots, \sigma_1)$ and "proof" polynomials $H_\nu(\mathbf{X}), \ldots, H_1(\mathbf{X})$. That is, for $1 \leq j \leq \nu$ and denoting $\boldsymbol{\sigma} = (\sigma_\nu, \ldots, \sigma_{j+1}$, the following equations hold:

$$p(\boldsymbol{\sigma}_{|j+1}, X_j, \ldots, X_1) - p(\boldsymbol{\sigma}_{|j+1}, \sigma_j, X_{j-1}, \ldots, X_1) = H_j(\mathbf{X})(X_j - \sigma_j)$$

Summing all the ν claims, we derive the PST verification equation. Note that the polynomial $H_j(\mathbf{X})$ is *independent* of the variables $X_\nu, \ldots, X_{j+1}$. Hence, each iteration is cheaper than the previous one.

The interesting part is that proofs for different positions *share elements*. Consider a polynomial $H(\mathbf{X})$ asserting $p(\sigma, X_\nu, \ldots, X_1) - p_\sigma(X_{\nu-1}, \ldots, X_1)$. This element will be part of the proof for all elements $\boldsymbol{\sigma} \in \Sigma^\nu$ whose first component is $\sigma_\nu = \sigma$. We utilize this fact to get a smaller amortized cost for evaluating all "proof" polynomials in the hypercube.

The tensor structure of the multivariate polynomial allows to express the openings in the hypercube as a tree. Each node of the tree corresponds to a partial evaluation of p. A proof polynomial H is associated with each of them. We demonstrate this in Fig. 5.5.

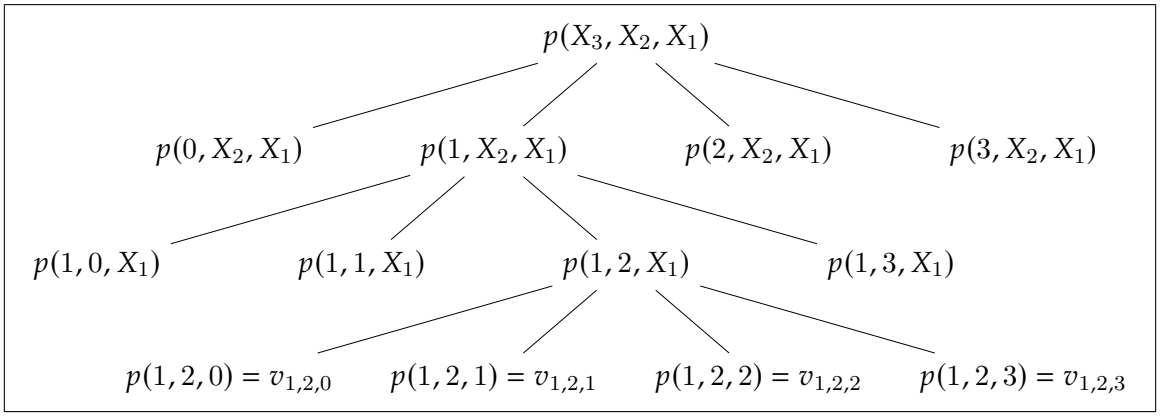


Figure 5.5: Tree structure for polynomials in 3 variables with individual degree at most 3. The dimension of the committed vectors with these parameters is $4^3 = 64$. We follow the path until we reach the leafs prefixed with $(1, 2)$. Note that each polynomial is an encoding of the leafs of the sub-tree it defines.

Finally, when using the Lagrange basis to encode polynomials, the interpolating polynomial $p(\mathbf{X})$ corresponding to $\mathbf{v}$ becomes $\boldsymbol{\lambda}(X_\nu, \ldots, X_1)^\top \mathbf{v}$. Furthermore, each node of the tree is of

the form $\boldsymbol{\lambda}(X_i, \ldots, X_1)^\top \mathbf{v}'$ where $\mathbf{v}'$ is the subvector of $\mathbf{v}$ corresponding to the leaf descendants of the node.

5.3.2 High Level Overview of the Construction

Tree structure. We can directly exploit the tree structure of the PST commitment scheme to pre-compute all proofs in quasi-linear time and simply hand them when needed. While this can be beneficial in some cases, it is an "extreme" approach. One might be willing to store *some* values and do *some* computation during opening of a position.

To achieve the flexible memory/time trade-off, we exploit the same tree-like structure, but instead of having the vector values in the leaves of the tree, we replace them with vector commitments using an arbitrary algebraic vector commitment scheme VCS. Concretely, the leaves contain elements $[\mathbf{r}]_1^\top \mathbf{v}_j$, where $[\mathbf{r}]_1 \in \mathbb{G}_1^k$ is the commitment key of VCS. To open a position of $\mathbf{v}$, we use the PST approach to reach corresponding leaf j, and then the opening algorithm of VCS on $\mathbf{v}_j$. The tree part can be efficiently pre-computed (memory resources) while the leaf part is opened on-demand (time resources).

We demonstrate the tree structure of our construction in Fig. 5.6. For simplicity, we consider $\Sigma = \{0, 1\}$ as the interpolating set. Each node can be consider as a succinct representation of the vectors encoded in the leafs of the sub-tree having the node as a root.

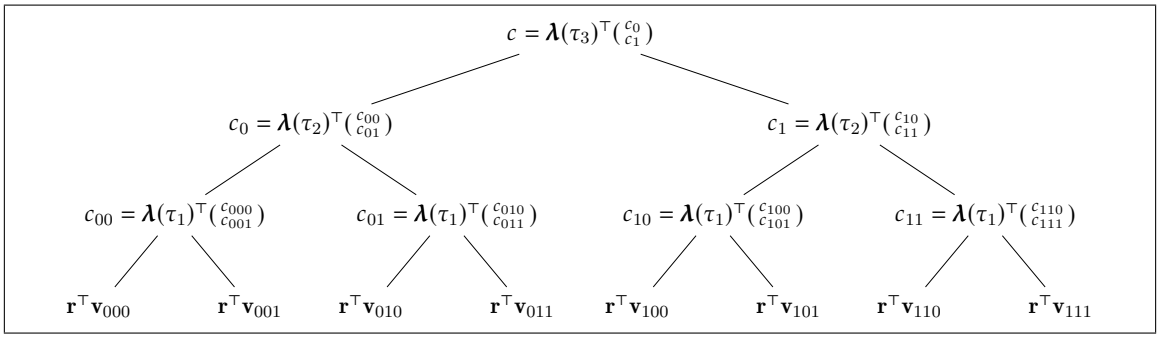


Figure 5.6: Demonstration of the tree structure of a commitment. Leaf nodes are commitments to k-dimensional vectors for an arbitrary vector commitment scheme. Each node is a commitment to its children under a Lagrange based key. The message space of the scheme is $k \cdot 2^3$.

The tree has a similar structure to Verkle trees [Kus18]. Each node can be considered as a succinct "commitment" to its children and proving involves (in some sense) sending this "commitments" and evidence they are well formed. The difference with Verkle trees is that the "commitment" does not satisfy a binding notion; indeed there are efficient ways to express each node in more than one ways by manipulating group elements.

Note that the root of the tree depends on the elements $\boldsymbol{\tau}, \mathbf{r}$. Viewing both $\boldsymbol{\tau} = (\tau_v, \ldots, \tau_1)$ and $\mathbf{r} = (r_k, \ldots, r_1)$ as formal variables $\mathbf{X}, \mathbf{R}$, we can treat the root node (the commitment) as an evaluation of a polynomial. Now, note that this polynomial corresponds to the interpolation

of the elements of the leaves in Σ^ν. Thus, the aforementioned polynomial is

$$p(\mathbf{X}, \mathbf{R}) = \boldsymbol{\lambda}(\mathbf{X})^\top (\mathbf{R}^\top \mathbf{v}_1, \ldots, \mathbf{R}^\top \mathbf{v}_{\ell^\nu}) = (\boldsymbol{\lambda}(\mathbf{X}) \otimes \mathbf{R})^\top \mathbf{v}$$

The prover can still evaluate one by one the variables $X_\nu, \ldots, X_1$ at $\sigma_\nu, \ldots, \sigma_1$ -as it would do in the simple PST case- and end up with a polynomial $q(\mathbf{R}) = p(\boldsymbol{\sigma}, \mathbf{R}) = \mathbf{R}^\top \mathbf{v}_j$. To ensure that q does not contain any X_j variable, we also include a low degree test in the proof. The evaluation of the latter polynomial at $[\mathbf{r}]_1$ corresponds to the leaf commitment at position $\boldsymbol{\sigma}$ and can be opened by employing the Open algorithm of the leaf commitment scheme with key $[\mathbf{r}]_1$.

Construction. First, we introduce some notation. Let $\Sigma \subseteq \mathbb{F}$ denote the interpolating set. Given $\boldsymbol{\sigma} = (\sigma_\nu, \ldots, \sigma_1) \in \Sigma^\nu$, we denote $\boldsymbol{\sigma}_{|i} = (\sigma_\nu, \ldots, \sigma_i) \in \Sigma^{\nu-i+1}$. For $\mathbf{v} = (\mathbf{v}_{\boldsymbol{\sigma}})_{\boldsymbol{\sigma} \in \Sigma^\nu}$ with $\mathbf{v}_{\boldsymbol{\sigma}} \in \mathbb{F}^k$ and $\boldsymbol{\sigma}_1 \in \Sigma^i$ we denote with $\mathbf{v}_{k,\boldsymbol{\sigma}_1}$ the vector $(\mathbf{v}_{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2})_{\boldsymbol{\sigma}_2 \in \Sigma^{\nu-i}}$, that is, the concatenation of vectors $\mathbf{v}_j$ whose m-ary representation of the index j is prefixed with $\boldsymbol{\sigma}_1$. Finally, we denote with $\boldsymbol{\tau}_{\nu,\ell}$ the ν-variate monomial basis of individual degree less than ν evaluated at $\tau_\nu, \ldots, \tau_1$. In all cases, we omit the subscript when it is clear from the context.

We present the construction next. We only consider the case of individual openings, but one can also consider openings of a larger family of linear functions of k-sized "blocks" associated with each leaf. The family is determined by the family of functions supported by the "leaf" commitment scheme.

Our construction is a linear vector commitment MVTree for vectors of dimension $k \cdot \ell^\nu$ that uses as a black box an algebraic vector commitment scheme VCS′ for vectors of dimension k. We index the opening positions as pairs (i, i') where i indexes the appropriate leaf corresponding to the position and i' the index corresponding to the position of the vector committed to the leaf. We present the construction in Fig. 5.7

We omit explicitly describing the update algorithm. Instead, we demonstrate in Thm. 33 how to efficiently update all proofs after modifying a position in the committed vector.

We summarize the properties of the construction in the following theorems.

Theorem 32. *Let* VCS′ *be an algebraic vector commitment scheme that satisfies completeness, homomorphic openings and position binding. Then,* MVTree *satisfies (1) completeness, (2) homomorphic openings and (3) position binding in the AGM under the* $(\ell - 1) \cdot \nu$-BSDH *assumption.*

Proof.

Completeness. Consider an honest execution of MVTree.Open. Let $y = \mathbf{v}_{(i,i')}$ for some $i = (\boldsymbol{\sigma})_\ell$. By inspection of the constructions and completeness of VCS′, the low degree test and the verification of the leaf opening succeed. It remains to show that the first test outputs 1.

Figure 5.7 MVTree construction based on the Lagrange multivariate polynomial basis.

$\mathsf{MVTree.KeyGen}(1^\kappa, n = k \cdot \ell^\nu) \to (\mathsf{pk}, \mathsf{vk})$**:**

$(\mathsf{pk}' = [\mathbf{r}]_1, \mathsf{vk}') \leftarrow \mathsf{VCS'.KeyGen}(1^\kappa, k)$

Let $\boldsymbol{\lambda}(X)$ be the vector of Lagrange polynomials associated to Σ

$\tau_\nu, \ldots, \tau_1 \leftarrow \mathbb{F}$

Output $\mathsf{pk} = (\mathsf{pk}', [\boldsymbol{\lambda}]_1 = [\boldsymbol{\lambda}(\tau_\nu) \otimes \cdots \otimes \boldsymbol{\lambda}(\tau_1) \otimes \mathbf{r}]_1, [\boldsymbol{\tau} \otimes \mathbf{r}]_1)$,

$\mathsf{vk} = (\mathsf{vk}', [\tau_\nu]_2, \ldots, [\tau_1]_2, [\tau_\nu^{\ell-1} \cdots \tau_1^{\ell-1}]_2)$

$\mathsf{uk} = (\{[\boldsymbol{\lambda}(\tau_\mathsf{j}) \otimes \cdots \otimes \boldsymbol{\lambda}(\tau_1) \otimes \mathbf{r}]_1\}_{\mathsf{j}=\nu-1}^{1})$,

$\mathsf{MVTree.Com}(\mathsf{pk}, \mathbf{v}) \to (\mathsf{c}, \mathsf{aux})$**:**

For all $\sigma \in \Sigma^\nu$: compute $(C_\sigma, \mathsf{aux}_\sigma) \leftarrow \mathsf{VCS'.Com}(\mathsf{pk}', \mathbf{v}_\sigma)$

Compute $\mathsf{c} = [p(\boldsymbol{\tau}, \mathbf{r})]_1 = [\boldsymbol{\lambda}]_1^\top \mathbf{v}$

Output $\mathsf{c}, \mathsf{aux} = \left(\{\mathsf{aux}_\sigma\}_{\sigma \in \Sigma^\nu}, \mathbf{v}\right)$

$\mathsf{MVTree.Open}(\mathsf{pk}, \mathsf{aux}, (i, i'), y) \to \pi$**:**

Let $i = (\sigma)_\ell$ in ℓ-ary.

Consider $\boldsymbol{\tau}, \mathbf{r}$ as formal variables $\mathbf{X} = (X_\nu, \ldots, X_1), \mathbf{R} = (R_k, \ldots, R_1)$.

Denote $p_{\nu+1}(\mathbf{X}, \mathbf{R}) = p(\mathbf{X}, \mathbf{R}) = (\boldsymbol{\lambda}(\mathbf{X}) \otimes \mathbf{R})^\top \mathbf{v}$

For all $\nu \geq j \geq 1$:

Compute $p_j(X_{j-1}, \ldots, X_1, \mathbf{R}) = \boldsymbol{\lambda}(X_{j-1}, \ldots, X_1, \mathbf{R})^\top \mathbf{v}_{\sigma_{|j}}$

Compute $H_j(X_j, \ldots, X_1, \mathbf{R})$ as

$$H_j(X_j, \ldots, X_1, \mathbf{R}) = \frac{p_{j+1}(X_j, \ldots, X_1, \mathbf{R}) - p_j(X_{j-1}, \ldots, X_1, \mathbf{R})}{(X_j - \sigma_j)}$$

Compute group element $[H_j]_1 = [H_j(\tau_j, \ldots, \tau_1, \mathbf{r})]_1$

Compute $\hat{\mathsf{c}}_\sigma = [\tau_\nu^{\ell-1} \cdots \tau_1^{\ell-1} \cdot \mathbf{r}]_1^\top \mathbf{v}_\sigma$

Compute $\pi' \leftarrow \mathsf{VCS'.Open}(\mathsf{pk}', \mathsf{aux}_\sigma, i', y)$

Output $\pi = ([H_\nu]_1, \ldots, [H_1]_1, \mathsf{c}_\sigma, \hat{\mathsf{c}}_\sigma, \pi')$

$\mathsf{MVTree.Verify}(\mathsf{vk}, \mathsf{c}, (i, i'), y, \pi) \to 0/1$**:**

Let $i = (\sigma)_\ell$ in ℓ-ary.

$b_{\mathsf{Path}} \leftarrow e\left(\mathsf{c} - \mathsf{c}_\sigma, [1]_2\right) = \sum_{j=1}^{\nu} e([H_j]_1, [\tau_j - \sigma_j]_2)$

$b_{\mathsf{LD\text{-}Test}} \leftarrow e\left(\mathsf{c}_\sigma, [\tau_\nu^{\ell-1} \cdots \tau_1^{\ell-1}]_2\right) = e\left(\hat{\mathsf{c}}_\sigma, [1]_2\right)$

$b_{\mathsf{Leaf}} \leftarrow \mathsf{VCS'.Verify}(\mathsf{vk}', \mathsf{c}_\sigma, i', y, \pi')$

Output $b_{\mathsf{Path}} \wedge b_{\mathsf{LD\text{-}Test}} \wedge b_{\mathsf{Leaf}}$

Let $p_{\nu+1}(X_\nu, \ldots, X_1, \mathbf{R}) = p(X_\nu, \ldots, X_1, \mathbf{R})$ be the polynomial $(\lambda(\mathbf{X}) \otimes \mathbf{R})^\top \mathbf{v}$. Next, consider the polynomial equations that the polynomials H_j are constructed to satisfy:

$$p_{j+1}(X_j, \ldots, X_1, \mathbf{R}) - p_j(X_{j-1}, \ldots, X_1, \mathbf{R}) = H_j(X_j, \ldots, X_1, \mathbf{R})(X_j - h_j)$$

Summing all these equation for $1 \leq j \leq \nu$ gives

$$p_{\nu+1}(X_\nu, \ldots, X_1, \mathbf{R}) - p_1 = \sum_{j=1}^{\nu} H_j(X_j, \ldots, X_1, \mathbf{R})(X_j - h_j)$$

and note that this corresponds to the verification equation. Thus, the first test passes. Finally, note that all the monomials involved these polynomials are included in the commitment key $\boldsymbol{\lambda}(\tau_\nu, \ldots, \tau_1) \otimes \mathbf{r}$, so the prover can encode these in $\mathbb{G}_1$.

Function Binding. First, we prove a claim stating that we can extract an opening of a leaf commitment in the AGM.

Claim 2. Let $\pi = ([H_\nu]_1, \ldots, [H_1]_1, \mathsf{c}_\sigma, \hat{\mathsf{c}}_\sigma, \pi')$ be an accepting proof. Then, for all algebraic adversaries $\mathcal{A}$ outputting accepting proofs, there exists an extractor that outputs opening of c_σ w.r.t. key $\mathbf{r}$ in the AGM.

Proof. Since we work in the AGM, we can extract coefficients $\hat{\mathbf{a}}, \mathbf{a}$ of polynomials $\hat{C}(\mathbf{X}, \mathbf{R})$, $C(\mathbf{X}, \mathbf{R})$ with degree less that $\ell - 1$ in $X_\nu, \ldots, X_1$ such that $[\hat{C}(\boldsymbol{\tau}, \mathbf{r})]_1 = \hat{\mathsf{c}}$ and $[C(\boldsymbol{\tau}, \mathbf{r})]_1 = \mathsf{c}$. By the low degree test, either $\hat{C}(\mathbf{X}, \mathbf{R}) = C(\mathbf{X}, \mathbf{R}) \cdot X_\nu^{\ell-1} \cdots X_1^{\ell-1}$ holds or $\hat{\mathbf{a}} \cdot (\boldsymbol{\tau}, \mathbf{r}) = \mathbf{a}(\boldsymbol{\tau}, \mathbf{r}) \cdot \tau_\nu^{\ell-1} \cdots \tau_1^{\ell-1}$. In the latter case, we find a non-trivial discrete logarithm relations of the elements of the commitment key. Assume the latter event did not happen. For the polynomial relation to hold with polynomial of degree less that $\ell - 1$ in $X_\nu, \ldots, X_1$, only the coefficients involving $\mathbf{R}$ are non-zero, in which case we extract a leaf commitment opening. □

Now, consider two opening/proof pairs c_σ $(i, i'), y_d, \pi_d$ for $d \in \{1, 2\}$ and let $i = (\boldsymbol{\sigma})_\ell$. We consider two cases. First, assume that $\mathsf{c}_{1,\sigma} = \mathsf{c}_{2,\sigma} = \mathsf{c}_\sigma$. By the fact that the low-degree test passes, we can extract an opening $\mathbf{v}_\sigma$ for this commitment except with negligible probability. Then, by the last verification test we have

$$\mathsf{VCS}'.\mathsf{Verify}(\mathsf{vk}', \mathsf{c}_\sigma, i', y_1, \pi_1') = \mathsf{VCS}'.\mathsf{Verify}(\mathsf{vk}', \mathsf{c}_\sigma, i', y_2, \pi_2') = 1$$

and since $y_1 \neq y_2$, we conclude that we have solved a position binding challenge for VCS'

Next, consider the case where $\mathsf{c}_{1,\sigma} \neq \mathsf{c}_{2,\sigma}$. We show that, in this case, an winning adversary can be used to break BSDH assumption. The reduction works as follows: on input $[1]_{1,2}$, $[\tau]_{1,2}, \ldots, [\tau^{(\ell-1)\cdot\nu}]_{1,2}$, sample a key for MVTree in the following way:

- Guess index $i = (\sigma)_\ell$.

- Sample $(\mathsf{pk}' = [\mathbf{r}]_1, \mathsf{vk}') \leftarrow \mathsf{VCS}'.\mathsf{KeyGen}(1^\kappa, k)$ along with the discrete logarithms of $[\mathbf{r}]_1$[7]

- for all $1 \le j \le \nu$ set $[\tau_j]_1 = [\rho_j \tau + \sigma_j]_1$ for random ρ_j.

- Compute the encodings of the multivariate Lagrange and monomial polynomials $[\boldsymbol{\lambda}(\tau_\nu, \ldots, \tau_1)]_1, [\boldsymbol{\tau}]_1$ and $[\tau_\nu]_2, \ldots, [\tau_1]_2, [\tau_\nu^{\ell-1} \cdot \tau_\nu^{\ell-1}]_2$. Note that this step is efficient since any element $\lambda_\sigma(\tau_1, \ldots, \tau_\nu)$ and in $\boldsymbol{\tau}$ is a polynomial of total degree at most $(\ell - 1) \cdot \nu$ on variables $\tau_j = \rho_j \tau$ so it can be computed using the ν powers of τ.

- Compute the proving commitment key by computing $\mathbf{r} \otimes [\boldsymbol{\lambda}(\tau_\nu, \ldots, \tau_1)]_1$.

First, we argue that the commitment key is correctly distributed. Indeed, we evaluate the multivariate Lagrange and monomial polynomials on a random point since $\tau, \rho_\nu, \ldots, \rho_1$ are uniformly distributed, and we compute $\mathbf{r}$ honestly.

Next, assume that the guess of index i was correct (which happens with $1/\ell^\nu = m/k$ probability) and that the verifying proofs contain $\mathsf{c}_{1,\sigma} \neq \mathsf{c}_{2,\sigma}$. By the fact that the low-degree test passes, we get two valid openings $\mathbf{v}_{1,\sigma}, \mathbf{v}_{2,\sigma}$ for these commitments w.r.t. the key $[\mathbf{r}]_1$. Since we know $\mathbf{r}$ in the field, we can compute the discrete logarithms of these elements: specifically,

$$(\mathsf{c}_{1,\sigma}, \mathsf{c}_{2,\sigma}) = (\mathbf{r}^\top \mathbf{v}_{1,\sigma}, \mathbf{r}^\top \mathbf{v}_{2,\sigma})$$

To simplify notation, denote these values $v, v' \in \mathbb{F}$ respectively and note that $v \neq v'$.

By the first verificatioon test, the following equations holds:

$$e\left([C - v]_1, [1]_2\right) = \sum_{j=1}^{\nu} e([H_j]_1, [\tau_j - \sigma_j]_2), e\left([C - v']_1, [1]_2\right) = \sum_{j=1}^{\nu} e([H'_j]_1, [\tau_j - \sigma_j]_2)$$

[7] We implicitly assume here that the distribution of the key generation algorithm is witness samplable. This is always the case for all distribution of interest.

Subtracting and setting $z_j = H_i - H'_i$ gives

$$e\left([v'-v]_1,[1]_2\right) = \sum_{j=1}^{\nu} e([z_j]_1,[\tau_j-\sigma_j]_2) \Leftrightarrow e\left([v'-v]_1,[1]_2\right) = \sum_{j=1}^{\nu} e([z_j]_1,\rho_j[\tau]_2) \Leftrightarrow$$

$$(v'-v)\cdot e\left([1]_1,[1]_2\right) = \tau \cdot e\left(\sum_{j=1}^{\nu}\rho_j[z_j]_1,[1]_2\right) \Leftrightarrow$$

$$\tau^{-1}\cdot e\left([1]_1,[1]_2\right) = (v'-v)^{-1}\cdot e\left(\sum_{j=1}^{\nu}\rho_j[z_j]_1,[1]_2\right) \Leftrightarrow$$

$$e\left([\tau^{-1}]_1,[1]_2\right) = \cdot e\left((v'-v)^{-1}\sum_{j=1}^{\nu}\rho_j[z_j]_1,[1]_2\right)$$

so by the final equation

$$\frac{1}{\tau^{-1}} e([1]_1,[1]_1) = (v'-v)^{-1}\sum_{j=1}^{\nu}\rho_j[z_j]_1$$

Therefore, $\left(0,(v'-v)^{-1}e(\sum_{j=1}^{\nu}\rho_j[z_j]_1,[1]_2)\right)$ is a solution to the BSDH challenge.

Homomorphic Proofs. Let $(\mathsf{c}_1,(i,i'),\mathbf{y}_1,\pi_1)$, $(\mathsf{c}_2,(i,i'),\mathbf{y}_2,\pi_2)$ be accepting statement-proof pairs with respect to some key vk. We show that for all $\alpha,\beta\in\mathbb{F}$, the statement-proof pair $(\mathsf{c},(i,i'),\mathbf{y},\pi) = (\alpha\mathsf{c}_1+\beta\mathsf{c}_2,(i,i'),\alpha\mathbf{y}_1+\beta\mathbf{y}_2,\alpha\pi_1+\beta\pi_2)$ is also accepting. Let

$$\pi = (\alpha[H_{\nu,1}]_1+\beta[H_{\nu,2}]_1,\ldots,\alpha[H_{1,1}]_1+\beta[H_{1,2}]_1,\\ \alpha\mathsf{c}_{1,\sigma}+\beta\mathsf{c}_{2,\sigma},\alpha\hat{\mathsf{c}}_{1,\sigma}+\beta\hat{\mathsf{c}}_{2,\sigma},\alpha\pi'_1+\beta\pi'_2)$$

be the combined proof. First, note that for i' corresponding to the leaf position, the second verification test $\mathsf{VCS}'.\mathsf{Verify}(\mathsf{vk}',\alpha\mathsf{c}_{1,\sigma}+\beta\mathsf{c}_{2,\sigma},i',\alpha\mathbf{y}_1+\beta\mathbf{y}_2,\alpha\pi'_1+\beta\pi'_2)$ outputs 1 by the homomorphic openings property of VCS'. For the first test, we have

$$\begin{aligned} e\left(\mathsf{c}-\mathsf{c}_\sigma,[1]_2\right) &= e\left(\alpha\mathsf{c}_1+\beta\mathsf{c}_2-\alpha\mathsf{c}_{1,\sigma}-\beta\mathsf{c}_{2,\sigma},[1]_2\right)\\ &= \alpha\cdot e\left(\mathsf{c}_1-\mathsf{c}_{1,\sigma},[1]_2\right)+\beta\cdot e\left(\mathsf{c}_2-\mathsf{c}_{2,\sigma},[1]_2\right)\\ &= \alpha\sum_{j=1}^{\nu} e([H_{j,1}]_1,[\tau_j-\sigma_j]_2)+\beta\sum_{j=1}^{\nu} e([H_{j,2}]_1,[\tau_j-\sigma_j]_2)\\ &= \sum_{j=1}^{\nu} e\left(\alpha[H_{j,1}]_1+\beta[H_{j,2}]_1,[\tau_j-\sigma_j]_2\right) = \sum_{j=1}^{\nu} e\left([H_j]_1,[\tau_j-\sigma_j]_2\right) \end{aligned}$$

Similarly, for the low degree test we have

$$\begin{aligned} e\left(\hat{\mathsf{c}}_{\sigma}, [1]_2\right) &= e\left(\alpha\hat{\mathsf{c}}_{1,\sigma} + \beta\hat{\mathsf{c}}_{2,\sigma}, [1]_2\right) = \alpha \cdot e\left(\hat{\mathsf{c}}_{1,\sigma}, [1]_2\right) + \beta \cdot e\left(\hat{\mathsf{c}}_{2,\sigma}, [1]_2\right) \\ &= \alpha \cdot e\left(\mathsf{c}_{1,\sigma}, [\tau_{\nu}^{\ell-1} \cdots \tau_1^{\ell-1}]_2\right) + \beta \cdot e(\mathsf{c}_{2,\sigma}, [\tau_{\nu}^{\ell-1} \cdots \tau_1^{\ell-1}]_2) \\ &= e\left(\alpha\mathsf{c}_{1,\sigma} + \beta\mathsf{c}_{2,\sigma}, [\tau_{\nu}^{\ell-1} \cdots \tau_1^{\ell-1}]_2\right) = e\left(\mathsf{c}_{\sigma}, [\tau_{\nu}^{\ell-1} \cdots \tau_1^{\ell-1}]_2\right) \end{aligned}$$

Thus, the new statement/proof pair passes all verification tests. □

Remark 2. Note that in the function binding proof, we only use the AGM to extract openings for the leaf commitments but not for the tree part of the construction. The latter is sound under falsifiable assumptions.

Theorem 33. *Consider construction* MVTree *and let* $\pi^{\sigma} = ([H_{\nu}^{\sigma}]_1, \ldots, [H_1^{\sigma}]_1, \mathsf{c}_{\sigma}, \hat{\mathsf{c}}_{\sigma}, \pi'_{\sigma})$ *be some proof of opening for a leaf commitment in position* $\boldsymbol{\sigma}$ *written in* ℓ*-ary. Then, computing all partial proofs* $\left\{([H_{\nu}^{\sigma}]_2, \ldots, [H_1^{\sigma}]_1, \mathsf{c}_{\sigma}, \hat{\mathsf{c}}_{\sigma})\right\}_{\sigma \in \Sigma^{\nu}}$ *can be done in* $O_s ecpar(k \cdot \nu \cdot \ell^{\nu}) = O_s ecpar(\nu \cdot m)$ *time and storing them needs* $O_s ecpar(\ell^{\nu}) = O_s ecpar(m/k)$ *space. Furthermore, if we update* C *by adding* δ *in some position* i^**, we can update all partial proofs in time* $O_s ecpar(\nu)$*.*

Proof.

Pre-computing partial proofs. Let $p(X_{\nu}, \ldots, X_1, \mathbf{R})$ be the polynomial encoding of $\mathbf{v}$ w.r.t. Σ^{ν} and consider the evaluation of polynomials $p_{\sigma_1}(\mathbf{X}, \mathbf{R}) = p(\sigma_1, \mathbf{X}, \mathbf{R})$ arranged in a tree: the root is the polynomial $p(X_{\nu}, \ldots, X_1, \mathbf{R})$ and the children of a node in level j are $\left\{p_{\sigma_{\nu},\ldots,\sigma_j}(\sigma, X_{j-2}, \ldots, X_1, \mathbf{R})\right\}_{\sigma \in \Sigma}$. Computing all proofs corresponds to computing a divisor polynomial for each node that asserts that the node is consistent with its parent node, plus some constant work for computing each leaf commitment along with its low degree proof. Assuming $\ell = O(1)$, each divisor polynomial proof can be computed in time linear in the total degree of $p_{\sigma_{\nu},\ldots,\sigma_j}$. A simple counting argument is enough to conclude the proof. In level j of the tree, we need to compute ℓ^j proofs, each for a polynomial of total degree $k \cdot \ell^{\nu-j}$. Thus, for each level of the tree, we need time linear in $k \cdot \ell^{\nu} = m$. Having ν levels, the total time is $O_{\lambda}(m \cdot \nu)$. For the space requirements, it is enough to note that the tree is has $O(m)$ nodes, and we associate one group element to each.

Updating all partial proofs. The updatability property follows directly by the homomorphic opening property of the construction. We focus on the computation needed for updating all stored proofs. The strategy is to consider the new commitment as $\mathsf{c}' = \mathsf{c} + \hat{\mathsf{c}}$ where $\hat{\mathsf{c}}$ is a commitment to the vector $\delta \cdot \mathbf{e}_{i^*}$, where $\boldsymbol{\sigma}$ denotes leaf corresponding to i^*. We claim that (1) we can compute all proofs for $\hat{\mathsf{c}}$ in logarithmic time and (2) all but $O(\nu)$ proof elements are 0. By this two facts the claim follows since we can combine all the non-zero proof elements of $\hat{\mathsf{c}}$ with the corresponding elements of c.

The commitment $\hat{c}$ corresponds to a polynomial of the form $p(\mathbf{X}, \mathbf{R}) = \delta \cdot \lambda_{\sigma}(\mathbf{X}) \cdot R_j$. All polynomials labeling nodes in the tree are 0 apart from the ones being in the path from the root to the leaf containing i^*. Such a node always has the zero polynomial nodes as descendants, and the proof corresponding to each is 0 since $0 = 0 \cdot (X_j - \sigma_j)$. The proof polynomials for the rest $\ell \cdot \nu$ nodes can be computed in constant time each and each can be encoded to the group in constant time since each involves a unique commitment key element. □

Efficiency. We only consider the case where $\ell = O(1)$. First, let's focus on the time needed to compute $[H_j]_1$. One can simply write the polynomial $p_j - p_{j-1}$ as a polynomial in $1, X_j, \ldots, X_j^{\ell-1}$ with polynomial coefficients in the other variables. Then, we can use standard (univariate) polynomial division to divide each term with $X_j - \sigma_j$ in constant time. To encode it in the group, it is enough to note that the total degree of each term is $k \cdot \ell^{j-1}$, so we need to perform ℓ multi-exponentiations of this size totaling in $O(k \cdot \ell^j)$ operations.

That said, we demonstrate the efficiency of the construction. The commitment key consists of linear in m group elements. Opening needs $O(k \cdot \ell^j)$ operations for each iteration, totaling in $O(k \cdot \ell^{\nu})$ time. By inspection of the construction, proofs size is $\log_{\ell}(m/k) + 2 + |\pi'|$, where π' is the size of an opening of the leaf commitment. Finally, verification consists of (1) a $\log_{\ell}(m/k)$-size pairing product equation, (2) a low degree test involving constant operations and (3) a verification of an opening of a leaf commitment.

Remark 3 (On aggregation)**.** The first two verification tests are pairing product equations. Assuming the leaf commitment verification is also a pairing product equation, one can use inner pairing products [BMM+21] to aggregate many such equations as done in [SCP+22] and, thus, achieve one-hop cross commitment aggregation. While the aggregated proof size decreases exponentially, this comes at the cost of a significant overhead for the prover due to the need to work in the target group. Reducing the proof size from $\log_2 m$ to roughly $\log_{\ell}(m/k)$ (assuming constant size/verification for leaf commitment opening) can make aggregation significantly cheaper for the prover.

Chapter 6

Folding Schemes with Selective Verification

This chapter is based on an unpublished paper that is joint work with Carla Ráfols.

Succinct non-interactive arguments of knowledge (SNARKs) have been proven an invaluable tool during the last decade, both in theoretical as well as practical terms. Such constructions allow a prover to convince a verifier that some NP relation is satisfied in a way such that communication and (in some cases) verification time are sublinear in the size of the NP witness. They can also be adapted to satisfy the zero-knowledge property, which guarantees that no information about the NP witness is leaked through the proof.

While the first real-world application of SNARKs [BCG+14b] aimed at preserving the privacy of the prover, the potential of this primitive for improving scalability in many applications is increasingly recognized, see for example roll-up architectures or the Filecoin network. In these applications, where the size of the computations is really large, the efficiency of the prover is the main bottleneck. Therefore, improving prover's efficiency is an active area of research [BCG20; BCL20; RR21; BCHO22].

These applications are a particular case of the problem of secure delegation of computation [GKR08], where an untrusted prover performs computations as a service to several "muggles", or computationally weak verifiers. The prover needs to convince the verifier (ideally with a non-interactive and publicly verifiable proof) of the correctness of the result, and verification should be much cheaper than performing the computation. SNARKs are a natural solution to this problem. However, the current cost of the prover is an obvious limitation in this scenario, as it directly translates into costs for the server and limits the possibilities of scaling the system.

Contributions. We aim to mitigate the necessity of large computational resources for the prover in applications where he provides services to many clients. Instead of trying to improve the efficiency of SNARK constructions, we take a different approach: we amortize the proving cost across multiple proofs of independent and unrelated statements. This means that, when having to make M computations of different statements, instead of producing M separate SNARK proofs for each, the prover "collapses" all these statements to a single statement in a verifiable way and only produces a proof for the latter using a SNARK. This is a novel application of *folding schemes* [KST21], originally introduced to improve recursive proof composition. We naturally extend this notion with a local property: testing only if a specific statement was considered during aggregation.

The guarantee we get is that if the proof for the aggregated statement verifies, then all statements are correct. Additionally, since the ultimate goal is to be able to prove unrelated statements, possibly coming from different parties, we augment aggregation with a property we call *selective verification*. This property captures that a small proof π_i -which importantly is sublinear in the number of aggregated statements- is evidence that a statement x_i was considered in the construction of the final aggregated statement and, thus, a proof for the latter along with π_i stands as a proof for the validity of x_i. Note that it is not necessary to even know the statements used in aggregation to assert the validity of x_i.

A crucial requirement for efficiency is that aggregation of M statements is more efficient than producing M SNARK proofs. We demonstrate this by considering natural aggregation schemes for various relations through simple public coin protocols and the Fiat-Shamir transform. Specifically, we consider (1) inner product relations of committed values, (2) vector commitment openings, (3) knowledge of openings of polynomial commitments, and (4) the relaxed R1CS relation of NOVA [KST21].

All the constructions are extremely efficient for the prover, who during folding does work comparable to reading the statements/witnesses (modulo a linear number of hash function computations needed to derive the non-interactive challenge of the Fiat-Shamir transform). Verification becomes a bit more expensive since now the verifier needs to assert, apart from the SNARK proof, that a statement in question is indeed "contained" in the aggregated statement. This is dominated by $\log M$ hash function computations where M is the number of aggregated statements.

Nevertheless, we argue that the construction is beneficial from the verifier's perspective as well. First, using a (simple) folding scheme, the verifier can "locally" aggregate many statements into a single statement x_i which will then be aggregated with other independent queries from other verifiers. Therefore, the additional cost of each verifier can be amortized when the verifier makes multiple queries. Second, since all verifiers need to assert the validity of the same folded statement, one could explore the possibility of distributing this task, incentivizing a few randomly chosen verifiers to check the aggregated statement. As long as one is honest, a cheating prover will be identified. If a verifier does not validate the proof himself, it can still query it in the future to the prover (along with other statements of interest that it locally

aggregates) instead of simply relying on other parties. Thus, we can fine-tune the verification cost on large scale systems without compromising security.

Our techniques are quite general. In particular, (1) we show a generic way to augment every non-interactive folding scheme with selective verification using combinatorial techniques, and (2) we do not rely on some specific SNARK construction, but can rather rely on any SNARK depending on the application's needs. Furthermore, constructing (simple) folding schemes is natural and easy using standard Σ-protocols, so the techniques can be easily adapted to work in a plethora of scenarios.

Folding schemes with selective verification can amortize the proving costs in large scale application where the prover should not be trusted. We demonstrate this using two examples:

1. delegation of computation as a service: a trustless proving server wants to "lend" its computational resources in the form of verifiably performing computations on behalf of clients, and

2. verifiable databases [BGV11]: a trustless database server stores, and manages databases for clients.

Our approach is very efficient when many verifiers wish to perform the same computation on different inputs. Performing arbitrary computations still incurs a significant overhead for the prover. Constructing folding schemes for arbitrary computations or exploring which types of computations can be aggregated efficiently using such objects in an interesting open question.

Related Work. Our techniques are inspired by a recent line of work on *proof composition techniques*, namely [BGH19; BCMS20; BCL+21; BDFG21]. In general, these techniques consider the notion of proof aggregation, namely, how to derive a single proof π that asserts the validity of two or more proofs. The motivation for this line of work is twofold. First, amortizing the cost of the (inefficient) verification of folding technique based constructions [BCC+16; BBB+18] and second, to construct proof carrying data [BCCT13] and incrementally verifiable computation [Val08].

Our technique differs in that (1) the main goal is to amortize the proving cost and (2) we consider the notion of aggregating unrelated statements, that is, one should assert the validity of statement without even knowing the other statements considered during aggregation. NOVA is closer to our work in that it directly considers aggregating statements instead of proofs, in an attempt to minimize the proving cost.

Perhaps closest to our work is [YLF+21]. There, they use a tree like structure similar to ours in order to derive the same Fiat-Shamir challenge across multiple parallel executions of an inner product argument protocol [BBB+18] with different parties. In particular, the protocol transcripts are committed in a Merkle tree so that each party can assert that its transcript was considered in the production of the challenge. We consider statement aggregation instead of executing multiple proofs in parallel which is conceptually different and more efficient.

6.1 Folding Schemes

In this section we recall the definition of *folding schemes* for NP relations introduced in NOVA [KST21]. On a high level, given an NP language $\mathcal{L}$ and the corresponding NP relation $\mathcal{R}$, a folding scheme allows a prover and a verifier to reduce the validity of 2 or more statements of the form $x_i \in \mathcal{L}$ to a single statement $x \in \mathcal{L}$. The resulting statement is of the same form, so it can be further aggregated. A prover knowing witnesses w_i s.t. $(x_i, w_i) \in \mathcal{R}$ for all the statements also obtains a witness w for the folded statement x.

A folding scheme takes to the extreme recent *proof composition techniques* used to construct PCD [BCCT13] and IVC [Val08]. Roughly, the core idea of these techniques is to aggregate proofs: given two statement/proof pairs, the prover and verifier reduce them to a single, different statement-proof pair that asserts the validity of both statements. Instead of verifying two proofs, a verifier needs only to verify (1) the aggregated proof and (2) aggregation was done correctly.

NOVA takes this approach to the extreme in the following sense: no proofs are considered anymore, the prover and verifier simply aggregate the statements themselves. Taking into account that producing proofs is a computationally intense task, this allows much better proving time with essentially no overhead for verification. Indeed, [KST21] introduces a folding scheme construction that captures all NP and allows very fast statement aggregation.

The formalization of a folding scheme is quite natural. In this work we only consider non-interactive folding schemes, since non-interactiveness is essential in the additional properties we introduce. Given a number of instance/witness pairs (x_i, w_i) that satisfy some NP relation, there exists a folding algorithm that outputs a new instance/witness pair (x, w) that also satisfies the NP relation, along with some evidence π that the new instance x is indeed an aggregated statement derived from the statements x_i. The properties required are:

1. *completeness*, stating that if we aggregate instance-witness pairs (x_i, w_i) satisfying the NP relation, then (1) folding results in an instance-witness pair also satisfying the relation and (2) the folding proof is accepted;

2. *knowledge soundness*, stating that if after correct aggregation the proving party knows a witness for the resulting statement, then it should also know witnesses for all statements (x_i, w_i) that were considered during aggregation.

Definition 37 (Folding scheme)**.** Let $\kappa \in \mathbb{N}$ be a security parameter and $\mathcal{L}_{\mathsf{par}}$ be an NP language parametrized by some parameters $\mathsf{par}(\kappa)$ depending on κ and $\mathcal{R}_{\mathsf{par}}$ the corresponding relation. Finally, let $M = \mathsf{poly}(\kappa)$. An M-folding scheme FS for the language family $\mathcal{L} = \{\mathcal{L}_{\mathsf{par}}\}_{\mathsf{par} \in \{0,1\}^*}$ is a tuple of an algorithms $\mathsf{FS} = (\mathsf{Fold}, \mathsf{FoldVrfy})$ such that for all $\mathsf{par} = \mathsf{par}(\kappa)$ and $m \leq M$

- $(x, w, \pi) \leftarrow \mathsf{Fold}(\mathsf{par}, x_1, w_1, \ldots, x_m, w_m)$: takes as input the parameters par, and m instance-witness pairs $(x_i, w_i) \in \mathcal{L}_{\mathsf{par}}$ and outputs a new instance-witness pair

$(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$ and a proof of correct folding π,

- $0/1 \leftarrow \mathsf{FoldVrfy}(\mathsf{par}, \mathsf{x}_1, \ldots, \mathsf{x}_m, \mathsf{x}, \pi)$: takes as input the parameters par, m instances x_i, an aggregated statement x and a proof of correct folding π and outputs a bit indicating whether folding was done correctly or not,

that satisfies the following properties:

1. **Completeness**: for all $m \leq M$, all $\mathsf{par} = \mathsf{par}(\kappa)$ and all (even computationally unbounded) algorithms $\mathcal{A}$,

$$\Pr\left[\begin{array}{c|r} \{\mathsf{q}_1, \ldots, \mathsf{q}_m\} \subseteq \mathcal{R}_{\mathsf{par}} \wedge & (\mathsf{x}_1, \mathsf{w}_1), \ldots, (\mathsf{x}_m, \mathsf{w}_m) \leftarrow \mathcal{A}(\mathsf{par}) \\ ((\mathsf{x}, \mathsf{w}) \notin \mathcal{R}_{\mathsf{par}} \ \vee \ b = 0) & \mathsf{q}_1 = (\mathsf{x}_1, \mathsf{w}_1), \ldots, \mathsf{q}_m = (\mathsf{x}_m, \mathsf{w}_m) \\ & (\mathsf{x}, \mathsf{w}, \pi) \leftarrow \mathsf{Fold}(\mathsf{par}, \mathbf{q}) \\ & b \leftarrow \mathsf{FoldVrfy}(\mathsf{par}, \mathbf{x}, \mathsf{x}, \pi) \end{array}\right] \leq \mathsf{negl}(\kappa)$$

2. **Knowledge soundness**: for all $m \leq M$ and all $\mathsf{par} = \mathsf{par}(\kappa)$ there exists a PPT extractor $\mathcal{E}$ such that for all PPT algorithms $\mathcal{A}$

$$\Pr\left[\begin{array}{c|r} (\mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathsf{par}} \ \wedge \ b = 1 & (\mathbf{x}, \mathsf{x}, \mathsf{w}, \pi) \leftarrow \mathcal{A}(\mathsf{par}) \\ \Rightarrow & \mathbf{w} \leftarrow \mathcal{E}^{\mathcal{A}}(\mathsf{par}) \\ \exists 1 \leq i \leq m \text{ s.t. } (\mathsf{x}_i, \mathsf{w}_i) \notin \mathcal{R}_{\mathsf{par}} & b \leftarrow \mathsf{FoldVrfy}(\mathsf{par}, \mathbf{x}, \mathsf{x}, \pi) \end{array}\right] \leq \mathsf{negl}(\kappa)$$

In Section 6.3 we present folding schemes for various relations: inner product relations of committed values, vector and polynomial commitment openings, the relaxed R1CS relation of [KST21]. We derive the constructions by means of public coin protocols that we compile to a non-interactive variant through the Fiat-Shamir heuristic.

6.2 Folding Schemes with Selective Verification

The main goal of this work to allow to reduce the resources used in "as a service" scenarios: a prover needs to serve multiple verifiers in a trustless way. A characteristic example is a prover that verifiably outsources its computational resources to verifiers who need to perform arbitrary computations.

Consider the case where a prover wants to serve m statements for m different parties. Simple folding is indeed a means to that goal: the prover needs to convince for the validity of a single statement to convince all verifiers about the validity of all m statements. Nevertheless, it is still inefficient in terms of verification. The inefficiency stems from the fact that in order to verify correct folding, all the statements need to be considered.

While this is natural in cases where a single verifier is interested in many statements, it can be prohibitive in scenarios where multiple verifiers are interested in the validity of different

statements: first, the verifiers need to know each others' queries to the prover to assert validity of the aggregated statement, and second, the verification cost scales linearly with the total number of statements considered.

In this section, we mitigate this issue by considering a stronger notion of folding schemes that allows to assert that a single statement was considered during aggregation of multiple statements -and hence knowledge of a witness of the latter implies knowledge of the witness of the former, without the need to know all the statements involved. Importantly, verification of inclusion of a single statement to the aggregated statement is *sublinear* in the total number of statements involved. We call this stronger notion folding with *selective verification*.

We first define the stronger notion of a folding scheme that supports *selective verification*. Then, we show that any non-interactive folding scheme can be compiled into one with selective verification.

Definition 38 (Folding scheme with selective verification)**.** Let $\kappa \in \mathbb{N}$ be a security parameter and $\mathcal{L}_{\mathsf{par}}$ be an NP language parametrized by some parameters $\mathsf{par}(\kappa)$ depending on κ and $\mathcal{R}_{\mathsf{par}}$ the corresponding relation. Finally, let $M = \mathsf{poly}(\kappa)$ and let $\mathsf{FS} = (\mathsf{Fold}, \mathsf{FoldVrfy})$ be an M-folding scheme for $\mathcal{L} = \left\{\mathcal{L}_{\mathsf{par}}\right\}_{\mathsf{par}\in\{0,1\}^*}$. FS has *selective verification* if there exists a pair of algorithms $(\mathsf{SelPrv}, \mathsf{SelVrfy})$ such that for all $m \leq M$

- $(\pi_1, \ldots, \pi_m) \leftarrow \mathsf{SelPrv}(\mathsf{par}, \mathsf{x}_1, \ldots, \mathsf{x}_m, \mathsf{x}, \pi)$: takes as input the parameters par, m instances $\mathsf{x}_1, \ldots, \mathsf{x}_m$, an aggregated instance x and the proof π produced by the algorithm Fold and outputs m proofs $\pi_1, \ldots, \pi_m$,
- $0/1 \leftarrow \mathsf{SelVrfy}(\mathsf{par}, \mathsf{x}, i, \mathsf{x}_i, \pi_i)$: takes as input the parameters par, an aggregated statement x, a position $i \in \{1, \ldots, m\}$, a statement x_i and a proof π_i and outputs a bit indicating if x_i was aggregated (among other statements) to x,

that satisfies the following properties:

1. **Selective completeness:** for all $m \leq M$, all $\mathsf{par} = \mathsf{par}(\kappa)$ and all (even computationally unbounded) algorithms $\mathcal{A}$,

$$\Pr\left[\begin{array}{c|r} \{\mathsf{q}_1, \ldots, \mathsf{q}_m\} \subseteq \mathcal{R}_{\mathsf{par}} \wedge & \mathsf{x}_1, \mathsf{w}_1, \ldots, \mathsf{x}_m, \mathsf{w}_m \leftarrow \mathcal{A}(\mathsf{par}) \\ \exists i \in \{1, \ldots, m\} : & \mathsf{q}_1 = (\mathsf{x}_1, \mathsf{w}_1), \ldots, \mathsf{q}_m = (\mathsf{x}_m, \mathsf{w}_m) \\ b_i = 0 & (\mathsf{x}, \mathsf{w}, \pi) \leftarrow \mathsf{Fold}(\mathsf{par}, \mathbf{q}) \\ & (\pi_1, \ldots, \pi_m) \leftarrow \mathsf{SelPrv}(\mathsf{par}, \mathbf{x}, \mathsf{x}, \pi) \\ & b_i \leftarrow \mathsf{SelVrfy}(\mathsf{par}, \mathsf{x}, i, \mathsf{x}_i, \pi_i) \end{array}\right] \leq \mathsf{negl}(\kappa)$$

2. **Selective knowledge soundness**: for all $m \leq M = \mathsf{poly}(\kappa)$ and all $\mathsf{par} = \mathsf{par}(\kappa)$ there exists a PPT extractor $\mathcal{E}$ such that for all PPT algorithms $\mathcal{A}$

$$\Pr\left[\begin{array}{c|r} (\mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathsf{par}} \ \wedge \ \mathsf{SelVrfy}(\mathsf{par}, \mathsf{x}, i, \mathsf{x}_i, \pi_i) = 1 & (i, \mathsf{x}_i, \pi_i, \mathsf{x}, \mathsf{w}) \leftarrow \mathcal{A}(\mathsf{par}) \\ \Rightarrow \ (\mathsf{x}_i, \mathsf{w}_i) \notin \mathcal{R}_{\mathsf{par}} & \mathsf{w}_i \leftarrow \mathcal{E}^{\mathcal{A}}(\mathsf{par}) \end{array}\right] \leq \mathsf{negl}(\kappa)$$

3. **Efficiency:** $|\pi_i| = o(m \cdot |\mathsf{x}|)$, namely, the proof size should be asymptotically smaller than the total size of aggregated statements.

Note that by completeness of the folding scheme, it should be the case the pair (x, w) also satisfies the relation $\mathcal{R}_{\mathsf{par}}$.

The definition captures that if (1) the prover knows a valid witness w for the aggregated statement x and (2) the i-th proof verifies, then it should be the case that the prover knows witness w_i such that $(\mathsf{x}_i, \mathsf{w}_i) \in \mathcal{R}_{\mathsf{par}}$. Note that from the perspective of a party asserting the validity of x_i, it is not necessary to know the other statements considered in the construction of x. Furthermore, the other statements need not be honestly generated; even if the adversary samples them, knowledge of the witness of the i-th statement is still guaranteed.

The efficiency condition rules out trivial constructions. Without it, one could set the proof of statement i to be simply the set of all aggregated statements along with a proof of correct folding. The verifier would then simply need to check that one of the statements corresponds to the one that is of interest to her. The interesting part of the definition is to achieve the same goal with *sublinear communication*.

Finally, note that we do not require the extractor to be able to extract all m statements that would explain the aggregated statement x; rather, we ask that given a witness for the aggregated statement and a valid proof, we can extract a witness only for the i-th statement. This is exactly what one would want for selective verification since ultimately, this is a *local property*: we want to ensure that some statement is correct without caring how we end up with an aggregated statement; the latter is simply a means to verify correctness of the statement of interest.

6.2.1 Construction of a Folding Scheme with Selective Verification

In this section we show how to achieve *selective verification* from any non-interactive folding scheme. We emphasize that since the folding schemes we present in this work are derived by applying the Fiat-Shamir heuristic to public coin protocols, we only achieve heuristic security.

The idea to achieve selective verification is quite simple. We leverage two facts: (1) that folding is *incremental*, meaning that a folded statement can be further folded and (2) that folding is non-interactive. These simple facts allow us to fold statements in a tree-like fashion so that the root node (the final aggregated statement) has small distance from the leafs (the statements to be folded) while it depends on all of them.

To convince of inclusion of the i-th statement in the folding process, the prover simply needs to give a series of incrementally folded statements that lead from the leaf to the root, similar to how a Merkle tree opening would be performed. To verify the process, the verifier asserts that each parent node is the folded statement derived from folding the children nodes using

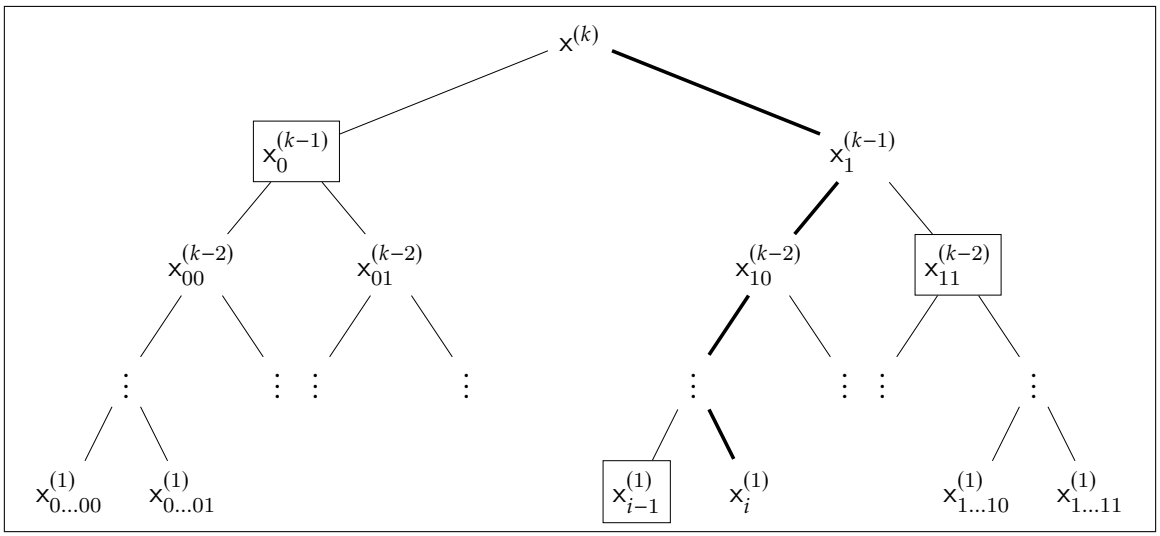


Figure 6.1: Demonstration of the process of deriving the folding tree. We assume we fold 2^k statements (the leafs of the tree). We index with the position of the node in the tree in binary and we use superscript for the level of the node in the tree. A node $x_{\mathbf{b}}^{(l)}$ is computed as the (non-interactive) folding of $x_{\mathbf{b}0}^{(l-1)}$ and $x_{\mathbf{b}1}^{(l-1)}$ using the underlying scheme FS. Bold edges denote the path the verification follows and rectangles the statements the prover presents to the verifier of statement i.

the proof of folding of the (non-selective) underlying folding scheme.

An important note is that if we have a statement of the form $x_1 \in \mathcal{L}$ and we are presented with a different statement $x_2 \in \mathcal{L}$, after folding these to a third statement $x \in \mathcal{L}$, knowledge of a witness for the latter ensures knowledge for both statements (in particular the first which is of interest to us) *even if the second is selected adversarially*. This means that from the perspective of a verifier interested in a specific statement, it is not important what other statements are considered or how they are sampled as long as they correctly end up to the claimed aggregated statement.

We first demonstrate the construction in Fig. 6.1. Next, we formally present the generic construction in Fig. 6.2. We start from a folding scheme FS and derive a folding scheme SFS with selective verification where the per-statement proof is logarithmic in size. Specifically, let FS be an M-folding scheme and any fixed constant $m \leq M$ (our constructions in this work will always consider folding $m = 2$ statements). Then, for any constant k, we derive an m^k-folding scheme with selective verification, where the proof size is $|\pi_i| = O(|x| \cdot k)$. This means we can aggregate polynomially many statements while each statement can be verified with a proof that is logarithmic in the number of statements.

An important observation, as far as efficiency is concerned, is that the proofs themselves are folded statements with their corresponding proofs, and thus yield little overhead to produce/verify the proofs -assuming the underlying folding scheme is concretely efficient. Essen-

tially, assuming that the underlying folding scheme is used for folding $m = 2$ statements, the prover has to perform $O(2^k)$ number of foldings and simply save the intermediate results in the process to be able to present as evidence later. As we will see in the next section, folding itself can be extremely efficient for many languages of interest. The overhead induced by folding for the prover is comparable to the time needed to simply read the statements! This can lead to significant improvements compared to -for example- producing a SNARK proof for each statement.

We next show that the construction SFS is a folding scheme (Thm. 34) and that it achieves selective verification (Thm. 35).

Theorem 34. *Let* FS *be an M-aggregation scheme for a language family $\mathcal{L}$ with corresponding relations $\mathcal{R}$. Then, for any constant $m \leq M$ and any constant $k \in \mathbb{N}$, construction* SFS *of Fig. 6.2 is an m^k-aggregation scheme for the same language family.*

Proof. Completeness follows directly by straightforward calculations and the completeness of SFS. We next show that SFS satisfies knowledge soundness.

Let $m' = m^k, \mathsf{x}_1, \ldots, \mathsf{x}_{m'}$ be statements and w a witness for the folded statement x output by an adversary $\mathcal{A}$. We construct an extractor $\mathcal{E}$ that extracts the witnesses $\mathsf{w}_1, \ldots, \mathsf{w}_{m'}$ given a witness for the folded statement w and a valid folding proof π, that uses as a black box the extractor $\mathcal{E}'$ for FS guaranteed to exist by knowledge soundness of FS.

Consider the m-ary tree defined by the honest SFS.FoldVrfy algorithm: the leafs are defined in the first level by the statements, that is, we label each leaf with $(\mathsf{x}_1^{(k)}, \perp), (\mathsf{x}_2^{(k)}, \perp), \ldots, (\mathsf{x}_{m'}^{(k)}, \perp)$ where $\mathsf{x}_j^{(k)} = \mathsf{x}_j$ and for each m-sized tuple of statements aggregated, we define a parent node connected to each of them labeled by the aggregated statement and the proof of correct aggregation. Note that verification passes, if

1. for any node labeled (x^*, π^*) with child nodes $(\mathsf{x}_1, \cdot), \ldots, (\mathsf{x}_m, \cdot)$ verification passes, namely, $\mathsf{FS.FoldVrfy}(\mathsf{par}, \mathsf{x}_1, \ldots, \mathsf{x}_m, \mathsf{x}^*, \pi^*) = 1$
2. the root node is labeled with $(\mathsf{x}, \cdot)$

We next show that for all such adversaries $\mathcal{A}$, there exists a family of extractors $\mathcal{E}_j^i$ for $1 \leq i \leq k-1$, $1 \leq j \leq m^i$ such that given as input a derived tree for some statements $\mathsf{x}_1, \ldots, \mathsf{x}_{m'}$, $\mathcal{E}_j^{(i)}$ extracts valid witnesses $\mathsf{w}_{j_1}^{(i+1)}, \ldots, \mathsf{w}_{j_m}^{(i+1)}$ for the statements $\mathsf{x}_{j_1}^{(i+1)}, \ldots, \mathsf{x}_{j_m}^{(i+1)}$ that are the children nodes of $\mathsf{x}_j^{(i)}$ in the derived tree. The construction is recursive. We denote $\mathcal{E}^{(0)}$ the trivial extractor that given the witness for the root node (output by the adversary $\mathcal{A}$), it simply outputs it.

Base case: $\mathcal{E}_1^{(1)}$ runs $\mathcal{E}^{(0)}$ to get the witness $\mathsf{w}_1^{(1)}$ for the root. It then queries the derived tree and constructs the adversary $\mathcal{A}_1^{(1)}$ that outputs $\mathsf{x}_1^{(2)}, \ldots, \mathsf{x}_m^{(2)}$, folded statement-witness

Figure 6.2 Construction of a folding scheme with selective verification from a plain folding scheme. Here we assume (w.l.o.g.) that the number of initial statements is m^k for some fixed constant m and some $k \in \mathbb{N}$.

$\mathsf{SFS.Fold}(\mathsf{par}, \mathsf{q}_1 = (\mathsf{x}_1, \mathsf{w}_1), \ldots, \mathsf{q}_{m'} = (\mathsf{x}_{m'}, \mathsf{w}_{m'}))$**:**

Let $m' = m^k$. If $k = 0$ then output $\mathsf{q}_1, \pi_1 = \perp$, otherwise, group the m^k statements to m groups of m^{k-1} elements each and denote them $\mathbf{q}_1, \ldots, \mathbf{q}_m$

for each $1 \le j \le m$ recursively compute $(\tilde{\mathsf{q}}_j = (\tilde{\mathsf{x}}_j, \tilde{\mathsf{w}}_j), \tilde{\pi}_j) \leftarrow \mathsf{SFS.Fold}(\mathsf{par}, \mathbf{q}_j)$

$(\mathsf{q}^* = (\mathsf{x}^*, \mathsf{w}^*), \pi^*) \leftarrow \mathsf{FS.Fold}(\mathsf{par}, \tilde{\mathsf{q}}_1, \ldots, \tilde{\mathsf{q}}_m)$

output $\mathsf{q}^*, \pi = (\pi^*, \tilde{\mathsf{x}}_1, \tilde{\pi}_1, \ldots, \tilde{\mathsf{x}}_m, \tilde{\pi}_m)$

$\mathsf{SFS.FoldVrfy}(\mathsf{par}, \mathsf{x}_1, \ldots, \mathsf{x}_{m'}, \mathsf{x}, \pi)$**:**

Let $m' = m^k$. If $k = 0$ then output 1 iff $\mathsf{x} = \mathsf{x}_1$, otherwise

1. group the m^k statements to m groups of m^{k-1} elements each and denote them $\mathbf{x}_1, \ldots, \mathbf{x}_m$
2. parse the proof as $\pi = (\pi^*, \tilde{\mathsf{x}}_1, \tilde{\pi}_1, \ldots, \tilde{\mathsf{x}}_m, \tilde{\pi}_m)$

for each $1 \le j \le m$ recursively compute $b_j \leftarrow \mathsf{SFS.FoldVrfy}(\mathsf{par}, \mathbf{x}_j, \tilde{\mathsf{x}}_j, \tilde{\pi}_j)$

$b \leftarrow \mathsf{FS.FoldVrfy}(\mathsf{par}, \tilde{\mathsf{x}}_1, \ldots, \tilde{\mathsf{x}}_m, \mathsf{x}, \pi^*)$

output $b \wedge b_1 \wedge \ldots \wedge b_m$

$\mathsf{SFS.SelPrv}(\mathsf{par}, \mathsf{x}_1, \ldots, \mathsf{x}_{m'}), \mathsf{x}, \pi)$**:**

Consider an execution of $\mathsf{SFS.FoldVrfy}(\mathsf{par}, \mathsf{x}_1, \ldots, \mathsf{x}_{m'}, \pi))$

Parse the tree defined by the former execution as follows:

- $(\mathsf{x}_i, \perp), \ldots, (\mathsf{x}_{m'}, \perp)$ are the leafs
- If (x^*, π^*) corresponds to the verification $b \leftarrow \mathsf{FS.FoldVrfy}(\mathsf{par}, \tilde{\mathsf{x}}_1, \ldots, \tilde{\mathsf{x}}_m, \mathsf{x}^*, \pi^*)$
  1. Add node $n = (\mathsf{x}^*, \pi^*)$
  2. Add edges from n to the nodes labeled with $(\tilde{\mathsf{x}}_i, \cdot)$

For $1 \le i \le m'$ set π_i the concatenation of the lables of the nodes corresponding to the path from $(x, \cdot)$ to $(x_i, \perp)$ with their sibling nodes.

Output $\pi_1, \ldots, \pi_{m'}$

$\mathsf{SFS.SelVrfy}(\mathsf{par}, \mathsf{x}, i, \mathsf{x}_i, \pi_i)$**:**

Let $i = (b_{k-1}, \ldots, b_0)$ in m-ary notation.

Parse $\pi_i = \left((\mathsf{x}^{(k)}, \pi^{(k)}), (\mathbf{x}^{(k-1)}, \pi^{(k-1)}), \ldots, (\mathbf{x}^{(2)}, \pi^{(2)}), (\mathbf{x}^{(1)}, \perp)\right)$

For each $k - 1 \le \ell \le 1$

Set $\mathsf{x}^{(\ell)}$ the b_ℓ-th element of $\mathbf{x}^{(\ell)}$

Output 0 if $\mathsf{FS.FoldVrfy}(\mathsf{par}, \mathbf{x}^{(\ell)}, \mathsf{x}_b^{(\ell+1)}, \pi^{(\ell+1)}) = 0$

Output 1 if $\mathsf{x} = \mathsf{x}^{(k)}$ and $\mathsf{x}^{(1)} = \mathsf{x}_i$.

pairs $\mathsf{x}_1^{(1)}, \mathsf{w}_1^{(1)}$ and proof $\pi^{(1)}$ which is part of the label of the root node. Finally, it invokes $\mathcal{E}'$ with access to $\mathcal{A}_1^{(1)}$ to derive witnesses $\mathsf{w}_1^{(2)}, \ldots, \mathsf{w}_m^{(2)}$ for the statements $\mathsf{x}_1^{(2)}, \ldots, \mathsf{x}_m^{(2)}$.

Recursive case: Now, let $i > 1$ and consider any j with $1 \leq j \leq m^i$. We construct an extractor $\mathcal{E}_j^{(i)}$ assuming the existence of an extractor for a level closer to the root node. Let $(\mathsf{x}_{p(j)}^{(i-1)}, \cdot)$ denote the label of the parent node of the node labeled with $(\mathsf{x}_j^{(i)}, \pi_j^{(i)})$ and let $(\mathsf{x}_{j_1}, \cdot), \ldots, (\mathsf{x}_{j_m}, \cdot)$ be the labels of the children of $\mathsf{x}_j^{(i)}$. Now, we construct $\mathcal{A}_j^{(i)}$ that has hardcoded the m-ary tree and works as follows:

- It invokes the extractor $\mathcal{E}_{p(j)}^{(i-1)}$ corresponding to statement $\mathsf{x}_{p(j)}^{(i-1)}$ to get a witness $\mathsf{w}_j^{(i)}$ for $\mathsf{x}_j^{(i)}$ (and all its siblings which it ignores).
- It then constructs an adversary $\mathcal{A}_j^{(i)}$ that simply outputs $\mathsf{x}_{j_1}, \ldots, \mathsf{x}_{j_m}$, the folded statement-witness pair $\mathsf{x}_j^{(i)}, \mathsf{w}_j^{(i)}$ and the proof of correct folding $\pi_j^{(i)}$ contained in the node label.
- Finally, it invokes the extractor $\mathcal{E}'$ of FS with access to $\mathcal{A}_j^{(i)}$ and gets witnesses w_{j_1}, $\ldots, \mathsf{w}_{j_m}$.
- It outputs witnesses $\mathsf{w}_{j_1}, \ldots, \mathsf{w}_{j_m}$.

We are now ready to construct the extractor $\mathcal{E}$. $\mathcal{E}$ queries $\mathcal{A}$ to get statements $\mathsf{x}_1, \ldots, \mathsf{x}_{m'}$, a folded statement-witness pair $(\mathsf{x}_1^{(1)}, \mathsf{w}_1^{(1)})$ and a proof of correct folding π. It then uses the proof and the statements to construct the tree, queries the extractors $\mathcal{E}_1^{(k-1)}, \ldots, \mathcal{E}_{m'/m}^{(k-1)}$ -each of which outputs m witnesses for m leaf nodes- and concatenates their outputs.

Let's now consider the running time and the probability of success of the extractor $\mathcal{E}$.

For the running time, let $t(\kappa, m)$ be the running time of $\mathcal{E}'$ and denote $t_i(\kappa, m)$ the running time of an extractor on level i (note that all these extractors are identical). By construction, we have that $t_i(\kappa, m) = t_{i-1}(\kappa, m) + t(\kappa, m)$ and $t_0(\kappa, m) = |\mathsf{w}|$, namely the time to output the folded witness w. This recurrence relation corresponds to $t_i(\kappa, m) = i \cdot t(\kappa, m) + |\mathsf{w}|$. Finally, the running time of the extractor $\mathcal{E}$ is

$$\begin{aligned} t_{\mathcal{E}}(\kappa, m, k) &= t_{\mathsf{SFS}}(\kappa, k, m) + m^{k-1} t_{k-1}(\kappa, m) = \\ &= t_{\mathsf{SFS}}(\kappa, k, m) + m^{k-1}(k-1) \cdot t(\kappa, m) + |\mathsf{w}| \end{aligned}$$

where $t_{\mathsf{SFS}}(k, m)$ is the time of SFS.FoldVrfy algorithm (equivalently the time needed to construct the statement tree). This corresponds to a quasilinear overhead $m' \log_m m'$ for the time of the extractor $\mathcal{E}$, which is polynomial for any number of polynomial statements.

We next show that the advantage of $\mathcal{E}$ is polynomially related to that of $\mathcal{E}'$. We denote with p' the probability that extractor $\mathcal{E}'$ succeeds in outputting the witnesses in FS conditioned

on $\mathcal{A}$ outputting a valid witness for the folded statement and a verifying proof, namely,

$$p' = \Pr\left[\begin{array}{l|r} \forall\, 1 \le i \le m\colon (\mathsf{x}_i, \mathsf{w}_i) \in \mathcal{R}_{\mathsf{par}} & (\mathsf{x}_1, \ldots, \mathsf{x}_m, \mathsf{x}, \mathsf{w}, \pi) \leftarrow \mathcal{A}(\mathsf{par}) \\ & (\mathsf{w}_1, \ldots, \mathsf{w}_m) \leftarrow \mathcal{E}'^{\mathcal{A}}(\mathsf{par}) \\ & \mathsf{FoldVrfy}\,(\mathsf{par}, \mathsf{x}_1 \ldots, \mathsf{x}_m, \mathsf{x}, \pi) = 1 \\ & (\mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathsf{par}} \end{array}\right]$$

Claim 3. Consider any adversary $\mathcal{A}$ against SFS and the folding tree derived by its output. Fix i, j such that $1 \le i \le k-1$ and $1 \le j \le m^i$ and consider the tree node $(\mathsf{x}_j^{(i)}, \pi_j^{(i)})$ and let $\mathsf{x}_1, \ldots, \mathsf{x}_m$ be its m children. Let W_i be the event that the extractor $\mathcal{E}_j^{(i)}$ outputs a valid witness for all the children nodes of $\mathsf{x}_j^{(i)}$, that is

$$W_i = \left\{\begin{array}{l|r} \forall 1 \le \ell \le m\colon (\mathsf{x}_\ell, \mathsf{w}_\ell) \in \mathcal{R}_{\mathsf{par}} & (\mathsf{x}_1, \ldots, \mathsf{x}_{m'}, \mathsf{x}, \mathsf{w}, \pi) \leftarrow \mathcal{A}(\mathsf{par}) \\ & (\mathsf{w}_1, \ldots, \mathsf{w}_m) \leftarrow \mathcal{E}_j^{(i)\,\mathcal{A}}(\mathsf{par}) \end{array}\right\}$$

Then $\Pr[W_i] \ge p' \Pr[W_{i-1}]$.

Proof. We have $\Pr[W_i] \ge \Pr[W_i \mid W_{i-1}] \Pr[W_{i-1}]$. Now, the probability of W_i conditioned on W_{i-1} is the probability that an extractor on the i-th level succeeds conditioned on the probability that the extractor on level $i-1$ succeeds. If the extractor of the parent node succeeds, then its output contains a valid statement/witness $\mathsf{x}_j^{(i)}, \mathsf{w}_j^{(i)}$ and therefore, $\mathcal{A}_i^j$ outputs a valid folded witness by construction. Thus, the probability of this event is exactly p'. □

Solving the recurrence relation gives that $\Pr[W_{k-1}] \ge p'^{k-2} \Pr[W_1]$. Now, $\Pr[W_1]$ is the probability that the extractor associated with the root node outputs valid witnesses assuming that $\mathcal{A}$ outputs a valid witness for the folded node. This means that, conditioned on $\mathcal{A}$ outputting a valid witness, $\Pr[W_{k-1}] \ge p'^{k-1}$.

Finally, consider the probability that $\mathcal{E}$ succeeds conditioned on $\mathcal{A}$ outputting a valid witness. This events happens if all extractors in level $k-1$ succeed. So, the probability that $\mathcal{E}$ fails is bounded by $\frac{m'}{m}(1 - p'^k) = m^{k-1}(1 - p'^{k-1})$. Noting that

$$1 - p'^{k-1} = (1 - p')(p'^{k-2} + \cdots + 1) \le (1 - p')(k-1)$$

we get for any adversaries $\mathcal{A}, \mathcal{A}'$ against knowledge soundness of SFS and FS respectively, $\mathsf{Adv}_{\mathcal{A}}(\kappa, m, k) \le (k-1)m^{k-1}\mathsf{Adv}_{\mathcal{A}'}(\kappa, m)$ □

We next show that the construction satisfies the stronger notion of selective verification. The proof is essentially identical; the only difference is that we simply focuses on a small part of the implicit tree which we construct using the elements contained in the proof for a single statement.

Theorem 35. *Let* SFS *be an M-aggregation scheme for a language family $\mathcal{L}$ with corresponding relations $\mathcal{R}$ Then, for any constant $m \leq M$ and any polynomial f, construction* SFS *of Fig. 6.2 satisfies selective verification.*

Proof. Assume (w.l.o.g.) that $f(m) = m' = m^k$ where m is an a priori fixed constant. Selective completeness follows directly by straightforward calculations and the completeness of SFS. Efficiency follows by the fact that a proof of inclusion of statement i contains $O(\log m')$ statements of $\mathcal{L}_{\mathsf{par}}$ and proofs of correct folding, which are polynomially related to the size of the statement. We next show that SFS satisfies selective knowledge soundness.

To simplify matters, we define the notion of the *derived ℓ-th subtree* defined by the proof, a statement x_ℓ and the folded statement. Concretely, we consider the subtree defined by the proof for the ℓ-th statement π_ℓ: it contains the part of the statement tree defined from the root to the leaf node $(\mathsf{x}_\ell, \perp)$ along with all the sibling nodes in the path.

Now let $(\ell, \mathsf{x}_\ell, \pi, \mathsf{x})$ and the derived subtree defined by these values. As in the previous proof, we construct recursively a series of extractors, one for each i with $1 \leq i \leq k-1$.

Base case: For $i = 1$, $\mathcal{E}_1^{(1)}$ runs $\mathcal{E}^{(0)}$ to get the witness $\mathsf{w}_1^{(1)}$ for the root. It then queries the derived subtree and constructs the adversary $\mathcal{A}_1^{(1)}$ that outputs $\mathsf{x}_1^{(2)}, \ldots, \mathsf{x}_m^{(2)}, \mathsf{x}^{(1)}, \mathsf{w}^{(1)}, \pi^{(1)}$. Finally, it invokes $\mathcal{E}'$ with access to $\mathcal{A}_1^{(1)}$ to get corresponding witnesses $\mathsf{w}_1^{(2)}, \ldots, \mathsf{w}_m^{(2)}$.

Recursive case: Let $\mathsf{x}^{(1)}, \ldots, \mathsf{x}^{(k)}$ be the statements contained in path from the root of the derived subtree to the leaf labeled with $(\mathsf{x}_i, \perp)$ and let $(\mathsf{x}_1^{(i+1)}, \cdot)$, …, $(\mathsf{x}_m^{(i+1)}, \cdot)$ be the labels of the children of $\mathsf{x}^{(i)}$. Now, we construct $\mathcal{A}^{(i)}$ that has hardcoded the derived subtree and works as follows:

- It invokes the extractor $\mathcal{E}^{(i-1)}$ corresponding to the parent statement $\mathsf{x}^{(i-1)}$ to get a witness $\mathsf{w}^{(i)}$ for statement $\mathsf{x}^{(i)}$ (and all its siblings which it ignores).
- It then constructs an adversary $\mathcal{A}^{(i)}$ that outputs $\mathsf{x}_1^{(i+1)}, \ldots, \mathsf{x}_m^{(i+1)}, \mathsf{x}^{(i)}, \mathsf{w}^{(i)}$ and the proof $\pi^{(i)}$.
- Finally, it invokes the extractor $\mathcal{E}'$ of FS with access to $\mathcal{A}^{(i)}$ and gets witnesses $\mathsf{w}_1^{(i+1)}$, …, $\mathsf{w}_m^{(i+1)}$ which it then outputs.

We then construct the extractor $\mathcal{E}$. $\mathcal{E}$ queries $\mathcal{A}$ to get $\ell, \mathsf{x}_\ell, \mathsf{x}, \pi$ and a witness for the folded statement $\mathsf{w}_1^{(1)}$. Then it simply queries $\mathcal{E}^{(k-1)}$ and outputs the witness corresponding to x_ℓ.

Working as in the proof of Thm. 34, we can deduce that the running time of the extractor is

$$t_{\mathcal{E}}(\kappa, m, k) = t_{\mathsf{SelVrfy}}(\kappa, k, m) + (k-1) \cdot t(\kappa, m) + |\mathsf{w}|$$

where $t_{\mathsf{SelVrfy}}(k, m)$ is the time of SFS.SelVrfy algorithm and $t(\kappa, m)$ is the time of the extractor $\mathcal{E}'$ of FS.

Finally, for the success probability of the extractor, it is enough to note that the proof verifies if the final folded statement computed during verification is the same as the claimed statement by the adversary $\mathcal{A}$, which means it is accompanied by a valid witness in the case of successful adversaries. Working as in the proof of Thm. 34 we get that for any adversaries $\mathcal{A}, \mathcal{A}'$ against selective knowledge soundness of SFS and knowledge soundness of FS respectively,

$$\mathsf{Adv}_{\mathcal{A}}(\kappa, m, k) \leq (k-1)\mathsf{Adv}_{\mathcal{A}'}(\kappa, m)$$

□

6.3 Folding Schemes from Interactive Public Coin Protocols

In this section we present folding schemes for various relations. We present four constructions:

1. a folding scheme for the language of inner product relations of committed values under algebraic commitments,
2. a folding scheme for the language of openings of vector commitments,
3. a folding scheme for the language of openings of extractable polynomial commitments and
4. a folding scheme for the relaxed R1CS relation of NOVA [KST21].

All the constructions are derived through simple public coin protocols. Thus, they can be compiled to non-interactive folding schemes through the Fiat-Shamir transform. Selective verifiability can then be achieved by means of the generic construction of Fig. 6.2. In all constructions we assume a base folding scheme for folding $m = 2$ statements. We start by introducing some notation for groups.

6.3.1 Folding Scheme for Inner Product Relation of Committed Values

Consider a language family $\mathcal{L}$ containing languages parametrized by a group key gk and two Pedersen commitment keys $[\mathbf{r}], [\mathbf{s}] \in \mathbb{G}^n$, each consisting of n uniformly distributed group elements.

The NP language is defined as

$$\mathcal{L}_{\mathsf{gk},[\mathbf{r}],[\mathbf{s}]} = \left\{([c], [d], z) \mid \exists \mathbf{a}, \mathbf{b} \text{ s.t. } [c] = [\mathbf{r}]^\top \mathbf{a},\ [d] = [\mathbf{s}]^\top \mathbf{b} \text{ and } z = \mathbf{a}^\top \mathbf{b}\right\}$$

Figure 6.3 Public coin protocol for folding statements for the language of inner product of openings of committed values. We include a final step where the prover sends the witness of the folded statement to the verifier.

for $i \in \{1,2\}$: $\mathsf{x}_i = ([c_i], [d_i], z_i)$, $\mathsf{w}_i = (\mathbf{a}_i, \mathbf{b}_i)$

$\mathcal{P}: \mathsf{q}_i = (\mathsf{x}_i, \mathsf{w}_i)$		$\mathcal{V}: \mathsf{x}_i$
$z_{1,2} = \mathbf{a}_1{}^\top \mathbf{b}_2, z_{2,1} = \mathbf{a}_2{}^\top \mathbf{b}_1$	$\xrightarrow{z_{1,2}, z_{2,1}}$	
	$\xleftarrow{\chi}$	$\chi \leftarrow \mathbb{F}$
$\mathbf{a} = \mathbf{a}_1 + \chi \mathbf{a}_2$		$[c] = [c_1] + \chi[c_2]$
$\mathbf{b} = \mathbf{b}_1 + \chi^2 \mathbf{b}_2$	$\xrightarrow{\mathbf{a}, \mathbf{b}}$	$[d] = [d_1] + \chi^2[d_2]$
		$z = z_1 + \chi \cdot z_{2,1} + \chi^2 \cdot z_{1,2} + \chi^3 z_2$
		$(([c], [d], z), \mathbf{a}, \mathbf{b}) \in_? \mathcal{R}_{\mathsf{gk},[\mathbf{r}],[\mathsf{s}]}$

and let $\mathcal{R}_{\mathsf{gk},[\mathbf{r}],[\mathsf{s}]}$ be the corresponding NP relation. We show how to fold two statements of this form to a single statement. Let

$$\mathsf{q}_1 = (([c_1], [d_1], z_1), (\mathbf{a}_1, \mathbf{b}_1)), \qquad \mathsf{q}_2 = (([c_2], [d_2], z_2), (\mathbf{a}_2, \mathbf{b}_2)),$$

such that (supposedly) $\mathsf{q}_1, \mathsf{q}_2 \in \mathcal{R}_{\mathsf{gk},[\mathbf{r}],[\mathsf{s}]}$. The strategy to fold the statements is as follows:

- The prover $\mathcal{P}$ first sends values $z_{1,2} = \mathbf{a}_1{}^\top \mathbf{b}_2$ and $z_{2,1} = \mathbf{a}_2{}^\top \mathbf{b}_1$.
- The verifier $\mathcal{V}$ then sends a random challenge $\chi \in \mathbb{F}$
- The prover and verifier construct the new statement $([c], [d], z)$ as

  $$[c] = [c_1] + \chi[c_2], \qquad [d] = [d_1] + \chi^2[d_2], \qquad z = z_1 + \chi z_{2,1} + \chi^2 z_{1,2} + \chi^3 z_2$$

  and the prover sets the new witness to $\mathbf{a} = \mathbf{a}_1 + \chi \mathbf{a}_2$, $\mathbf{b} = \mathbf{b}_1 + \chi^2 \mathbf{b}_2$.

It is easy to assert that the new witness pair satisfies the NP relation as long as the two initials statement do. Intuitively, this satisfies soundness since (1) a prover being able to open a commitment of the form $[\alpha] + \chi[\beta]$ for a random χ should in fact know openings for the combined commitments since they are defined before the challenge χ and (2) the "mixed" inner products $z_{1,2}, z_{2,1}$ are defined before the challenge χ is known, which means that one could treat the resulting relation as a polynomial relation on a formal variable X, that is $\mathbf{a}(X)^\top \mathbf{b}(X) = z(X)$. If this relation holds formally, then it is to assert that both $\mathbf{a}_1{}^\top \mathbf{b}_1 = z_1$ and $\mathbf{a}_2{}^\top \mathbf{b}_2 = z_2$ hold. The challenge essentially is a randomized test on this relation.

We define the protocol formally in Fig. 6.3. Note that contrary to what happens in the folding technique, we use u Next, we show that (1) an honest prover always outputs a valid statement-witness pair, and (2) given an adversary that outputs a valid witness after the execution of

the protocol for the folded statement, we can extract witnesses for the two statements $\mathsf{x}_1, \mathsf{x}_2$. Note, that if this holds, the Fiat-Shamir compiled construction directly yields a non-interactive folding scheme, where the proof is simply the pair of elements $z_{1,2}, z_{2,1}$ sent from the prover to the verifier.

Theorem 36. *Consider construction of fig. 6.3. Then the following conditions hold:*

1. *The resulting statement-witness pair defined after the end of the protocol satisfies the NP relation $\mathcal{R}_{\mathsf{gk},[\mathbf{r}],[\mathbf{s}]}$ and*
2. *The protocol satisfies special-soundness, namely, given four accepting executions for distinct verifier challenges, we can extract witnesses $\mathsf{w}_1, \mathsf{w}_2$ for the initial statements $\mathsf{x}_1, \mathsf{x}_2$ except with negligible probability.*

Proof.

1. We simply need to verify the NP relation is satisfied. First, we check that the openings of the commitments are valid. We have

$$[\mathbf{r}]^\top(\mathbf{a}_1 + \chi\mathbf{a}_2) = [\mathbf{r}]^\top\mathbf{a}_1 + \chi[\mathbf{r}]^\top\mathbf{a}_2 = [c_1] + \chi[c_2] = [c]$$
$$[\mathbf{s}]^\top(\mathbf{b}_1 + \chi^2\mathbf{b}_2) = [\mathbf{s}]^\top\mathbf{b}_1 + \chi^2[\mathbf{s}]^\top\mathbf{b}_2 = [d_1] + \chi^2[d_2] = [d]$$

   Finally, we assert that the inner product is correct. We have

$$\begin{aligned}\mathbf{a}^\top\mathbf{b} &= (\mathbf{a}_1 + \chi\mathbf{a}_2)^\top(\mathbf{b}_1 + \chi^2\mathbf{b}_2)\\ &= \mathbf{a}_1{}^\top\mathbf{b}_1 + \chi^2\mathbf{a}_1{}^\top\mathbf{b}_2 + \chi\mathbf{a}_2{}^\top\mathbf{b}_1 + \chi^3\mathbf{a}_2{}^\top\mathbf{b}_2\\ &= z_1 + \chi^2 z_{1,2} + \chi z_{2,1} + \chi^3 z_2 = z\end{aligned}$$

2. Assume we have four accepting executions of the interactive protocol with different challenges $\chi_1, \chi_2, \chi_3, \chi_4$. First we show that using any two transcripts we can extract valid openings for the commitments $[c_1], [d_1], [c_2], [d_2]$. We first focus on the commitments $[c_1], [c_2]$. After successful execution with challenges χ_i, χ_j, we have two openings $\mathbf{a}^{(i)}, \mathbf{a}^{(j)}$ for commitments $[c^{(i)}] = [c_1] + \chi_i[c_2]$ and $[c^{(j)}] = [c_1] + \chi_j[c_2]$ respectively. This means that

$$[c_1] + \chi_i[c_2] = [\mathbf{r}]^\top\mathbf{a}^{(i)}, \qquad [c_1] + \chi_j[c_2] = [\mathbf{r}]^\top\mathbf{a}^{(j)}$$

   Denote with $\mathbf{X}_{i,j}$ the matrix whose first row is $(1, \chi_i)$ and second row is $(1, \chi_j)$ and note that this matrix is invertible for $\chi_i \neq \chi_j$. We can write the above system of equations as

$$\mathbf{X}_{i,j}\begin{bmatrix}c_1\\ c_2\end{bmatrix} = [\mathbf{r}]^\top\begin{pmatrix}\mathbf{a}^{(i)}\\ \mathbf{a}^{(j)}\end{pmatrix}$$

   Denoting $\mathbf{X}_{i,j}^{-1}$ the inverse of $\mathbf{X}_{i,j}$ we get

$$\begin{bmatrix}c_1\\ c_2\end{bmatrix} = [\mathbf{r}]^\top\mathbf{X}_{i,j}^{-1}\begin{pmatrix}\mathbf{a}^{(i)}\\ \mathbf{a}^{(j)}\end{pmatrix}$$

so we indeed extract openings for the two commitments. Furthermore, note for any pair $i \neq j$ with $i, j \in \{1, 2, 3, 4\}$ we extract the same openings $\mathbf{a}_1, \mathbf{a}_2$ except with negligible probability, otherwise we break the binding property of the commitment scheme. Similarly, we extract openings $\mathbf{b}_1, \mathbf{b}_2$ for the commitments $[d_1], [d_2]$. Now, since we have an accepting witness for each of the four executions the following equations hold:

$$\mathbf{a}^{(i)\top}\mathbf{b}^{(i)} = z_1 + \chi_i^2 z_{1,2} + \chi_i z_{2,1} + \chi_i^3 z_2, \qquad 1 \leq i \leq 4$$

Assuming that no breaking of the binding property has happened, each opening $\mathbf{a}^{(i)}$ can be written as $\mathbf{a}^{(i)} = \mathbf{a}_1 + \chi_i \mathbf{a}_2$ for the same $\mathbf{a}_1, \mathbf{a}_2$ and similarly for the $[d_1], [d_2]$ commitments. We can now rewrite the above equations as

$$(\mathbf{a}_1 + \chi_i \mathbf{a}_2)^\top (\mathbf{b}_1 + \chi_i^2 \mathbf{b}_2) = z_1 + \chi_i^2 z_{1,2} + \chi_i z_{2,1} + \chi_i^3 z_2$$

or equivalently

$$\mathbf{a}_1{}^\top \mathbf{b}_1 + \chi_i \mathbf{a}_2{}^\top \mathbf{b}_1 + \chi_i^2 \mathbf{a}_1{}^\top \mathbf{b}_2 + \chi_i^{3\top} \mathbf{a}_2{}^\top \mathbf{b}_2 = z_1 + \chi_i^2 z_{1,2} + \chi_i z_{2,1} + \chi_i^3 z_2$$

Viewing this as a polynomial equation of degree 3 and noting it is satisfied for 4 distinct points, it should hold as a polynomial identity, therefore $\mathbf{a}_1{}^\top \mathbf{b}_1 = z_1$ and $\mathbf{a}_2{}^\top \mathbf{b}_2 = z_2$.

□

Efficiency. The work of the prover consists of a *linear number of field operations*, specifically, combining the two witness with the random challenge of the verifier χ and computing the cross term inner products $z_{1,2}$ and $z_{2,1}$. The verifier performs a constant number of operations in the field and group to derive the new statement. In the context of non-interactive folding with selective verification, folding M statements of size n consists of $O(Mn)$ field operations and $O(M)$ hash computations for the prover and $O(\log M)$ field and group operations and hash computations for the verifier.

6.3.2 Folding Scheme for Vector Commitment Openings

A vector commitment [CF13] allows a prover to succinctly commit to a vector $\mathbf{a} \in \mathbb{F}^n$ and later verifiably open a subset $S \subseteq \{1, \ldots, n\}$ of the positions of the committed vector. We construct a folding scheme for the language of openings of algebraic vector commitments. Recall that an algebraic commitment is any "Pedersen" type commitment scheme, that is, the commitment key is sampled as $\mathbf{r} \leftarrow \mathcal{D}_{1,n}$, where $\mathcal{D}_{1,n}$ is a matrix distribution and committing to $\mathbf{a}$ is done by computing $[\mathbf{r}]_1^\top \mathbf{a}$[1]. In what follows, we denote with $\mathbf{a}_{|S}$ the subvector of $\mathbf{a}$ defined by the set $S \subseteq \{1, \ldots, n\}$ More concretely, we consider the language

$$\mathcal{L}_{\mathsf{gk},[\mathbf{r}]} = \left\{([c], S, \mathbf{a}_S) \mid \exists \mathbf{a} \text{ s.t. } [c] = [\mathbf{r}]^\top \mathbf{a} \text{ and } \mathbf{a}_{|S} = \mathbf{a}_S\right\}$$

[1]The key does not need necessarily to be a vector. For more involved applications, a matrix $\mathbf{R}$ is sampled and the commitment is computed as $\mathbf{R}^\top \mathbf{a}$.

Our strategy for constructing a folding scheme for this relation is to reduce it to an inner product. That is, we first show that the language above can be interactively reduced to an inner product statement, and then we can use the folding scheme of the previous section for inner product relations.

The reduction is quite simple. We first note that the validity of an S-subopening can be expressed as $|S|$ inner products: for each $s \in S$ we need to assert that $\mathbf{a}^\top \mathbf{e}_{n,s} = a_s$, where $\mathbf{e}_{n,s}$ is the n-dimensional vector which is 0 everywhere except the s-th condition. These $|S|$ statements can be compressed to a single inner product by taking a random linear combination of the equations. That is, consider a vector $\mathbf{b}$ that is uniformly distributed conditioned on $b_i = 0$ for all $i \in \{1, \ldots, n\} \setminus S$. Then, with overwhelming probability the relation $\mathbf{a}^\top \mathbf{b} = \mathbf{a}_S^\top \mathbf{b}$ holds. To reduce communication, we do not use a uniform $\mathbf{b}$, we use a pseudorandom one: the monomials of a random element $\chi \in \mathbb{F}$. This is a well-known technique that reduces many inner products to a single "twisted" instance, as in [BCC+16].

We can now express the above as an instance of an inner product relation. Let $[c]$ be some committed value and $\mathbf{a}_S$ a claimed opening at positions S. The verifier reduces this claim to an inner product by doing the following:

- It samples $\chi \leftarrow \mathbb{F}$ and constructs the vector $\mathbf{b} = (1, \chi, \ldots, \chi^{|S|-1})$
- It commits to the vector $\mathbf{b}$ as $[d] = [\mathbf{r}_{|S}]^\top \mathbf{b}$. Note that this corresponds to a commitment w.r.t. $[\mathbf{r}]$ which is 0 everywhere outside S.
- It computes the inner product $z = \mathbf{a}_S^\top \mathbf{b}$.
- It sends χ to the prover and asks to prove the IP statement $([c], [d], z)$

A simple application of the Schwartz-Zippel lemma is enough to assert that $([c], [d], z)$ is a valid inner product statement if and only if the S opening of $[c]$ is $\mathbf{a}_S$ (except with negligible probability).

We present the interactive reduction of VC opening to inner product in Fig. 6.4. After applying the reduction we can simply fold the reduced statement with other IP statements.

Theorem 37. *Consider construction of Fig. 6.4.*

1. *The resulting statement-witness pair defined after the end of the protocol satisfies the inner product NP relation* $\mathcal{R}_{\mathsf{gk},[\mathbf{r}],[\mathbf{r}]}$*, and*
2. *The protocol satisfies special-soundness, namely, given* $|S|$ *valid statement-witness pairs after distinct verifier challenges, we can extract a valid witness* w *for the initial statement* x *except with negligible probability.*

Proof.

Figure 6.4 Public coin protocol for interactively reducing a VC opening claim to an inner product claim.

$\mathsf{x} = ([c], S, \mathbf{a}_S), \mathsf{w} = \mathbf{a}$

$\mathcal{P}: \mathsf{q} = (\mathsf{x}, \mathsf{w})$		$\mathcal{V}: \mathsf{x}_i$
	χ $\longleftarrow$	$\chi \leftarrow \mathbb{F}$
$\mathbf{b} = (1, \chi, \ldots, \chi^{\lvert S\rvert-1})$		$\mathbf{b} = (1, \chi, \ldots, \chi^{\lvert S\rvert-1})$
$[d] = [\mathbf{r}_{\lvert S}]^\top \mathbf{b}$		$[d] = [\mathbf{r}_{\lvert S}]^\top \mathbf{b}$
$z = \mathbf{a}_S^\top \mathbf{b}$		$z = \mathbf{a}_S^\top \mathbf{b}$

$\mathsf{x}' = ([c], [d], z), \mathsf{w}' = \mathbf{a}, \mathbf{b}$

1. Let $\mathbf{b}'$ be the vector that agrees with b on S and is zero everywhere in $\{1, \ldots, n\} \setminus S$ and note this corresponds to an opening of $[d]$. We have

$$z = \mathbf{a}^\top \mathbf{b}' = \sum_{i=1}^{n} a_i b'_i = \sum_{s \in S} a_s b'_s = \sum_{s \in S} a_s b_s = \mathbf{a}_S^\top \mathbf{b}$$

2. After each execution, we get a valid opening $\mathbf{a}$ for $[c]$. All these openings should be the same except with negligible probability, otherwise we break the binding property of the vector commitment. We next show that $\mathbf{a}_S = \mathbf{a}_{|S}$. Since each inner product is valid, the following relation is satisfied for each execution $\mathbf{a}_{|S}\mathbf{b} = \mathbf{a}_S\mathbf{b}$.

   Equivalently, we have $(\mathbf{a}_{|S} - \mathbf{a}_S)^\top \mathbf{b} = 0$. Next, note that, since each $\mathbf{b}$ encodes monomials of degree bounded by $|S| - 1$ derived from some field element χ, this corresponds to $|S|$ polynomial relations of the form $p(\chi) = 0$. Since p's degree is bounded by $|S| - 1$ and it has $|S|$ roots, it should be the case that p is identically zero, which means that $(\mathbf{a}_{|S} = \mathbf{a}_S)$.

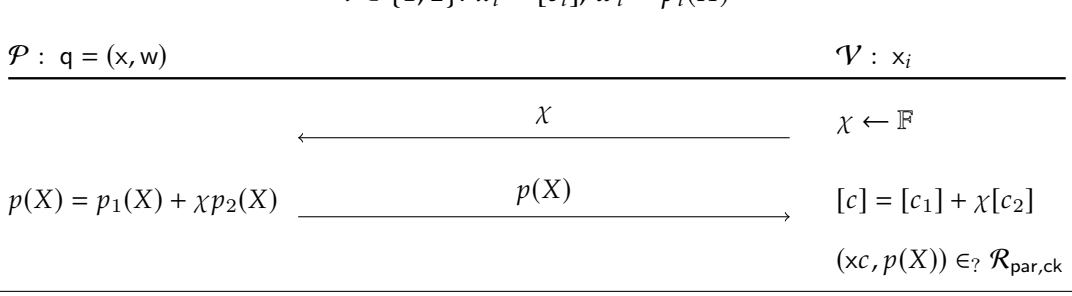
□

The only efficiency overhead for both the prover and the verifier is to compute the values $[d], z$ each needing $|S|$ field and group operations.

6.3.3 Folding Scheme for Polynomial Commitment Openings

A polynomial commitment scheme [KZG10] is a primitive that allows a prover to succinctly commit to a polynomial and later open it at an arbitrary point. Our next construction allows to fold statements about openings of polynomial commitments. We assume a linearly homomorphic polynomial commitment, namely if $[c], [d]$ are commitments to $p(X), q(X)$, then $\alpha[c] + \beta[d]$ is a commitment to $\alpha p(X) + \beta q(X)$. The language is parametrized by parameters and a key for a polynomial commitment scheme $\mathsf{par}, \mathsf{ck}$. We assume that all polynomials are

Figure 6.5 Public coin protocol for folding statements for the openings of polynomial commitments. We include a final step where the prover sends the witness of the folded statement to the verifier.



of a fixed degree d; generalizing this to achieve any degree $d \leq D$ for some bound D and hiding commitments is also possible. Formally, the language is defined as

$$\mathcal{L}_{par,ck} = \left\{[c] \mid \exists p(X) \text{ s.t. } p(X) \in \mathbb{F}[X] \text{ of degree d and } [c] = \mathsf{Com}_{par,ck}(p(X))\right\}.$$

The construction simply consists of combining the two polynomial commitments with a random challenge from the verifier. We present the construction in Fig. 6.5. We present a theorem capturing the properties of the protocol next.

Theorem 38. *Consider construction of Fig. 6.5. Then the following conditions hold:*

1. *The resulting statement-witness pair defined after the end of the protocol satisfies the NP relation $\mathcal{R}_{par,ck}$, and*
2. *The protocol satisfies special-soundness, namely, given two accepting executions for distinct verifier challenges, we can extract witnesses w_1, w_2 for the initial statements x_1, x_2 except with negligible probability.*

Proof.

1. This follows directly by the homomorphic properties of the commitment scheme.
2. For special soundness, it is enough to note that given two valid transcripts for different challenges, we can solve a simple system of polynomial equations $p^{(1)}(X) = p_1(X) + \chi_1 p_2(X)$, $p^{(2)}(X) = p_1(X) + \chi_2 p_2(X)$ to extract polynomials $p_1(X), p_2(X)$ that are valid openings for $[c_1], [c_2]$, respectively.

□

Efficiency. In this construction, the proof of correct folding is trivial: the challenge χ fully defines the aggregated statement and witness pair. The work of the prover and verifier consists

of a linear number of field operations and a constant number of group operations, respectively. In the context of non-interactive folding with selective verification, aggregating M statements of size n is dominated by $O(Md)$ field operations and $O(M)$ hash computations for the prover and $O(\log M)$ group operations and hash computations for the verifier.

6.3.4 Folding Scheme for Committed Relaxed R1CS

NOVA [KST21] introduces a novel variant of the R1CS characterization of NP, called *relaxed R1CS* which is amenable to folding, that is, there exists an efficient folding scheme for this language. The protocol they present is a public coin one and can be compiled to a non-interactive one by using the Fiat-Shamir heuristic.

The limitation of the constructions stems from the fact that the two initial instances have to describe the same computation. Nevertheless, the resulting folding scheme is very efficient both for the prover and the verifier. Thus, compiling this to a non-interactive aggregation scheme with selective verification allows a prover to efficiently convince multiple verifiers about different statements w.r.t. the same computation.

We describe the language of committed relaxed R1CS. The language is parametrized by

1. parameters for the relaxed R1CS instance $(\mathbf{A}, \mathbf{B}, \mathbf{C}, m, n, \ell)$ where $m, n, \ell \in \mathbb{F}$ and $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{F}^{m \times m}$, each having at most n non-zero entries,

2. parameters for a Pedersen commitment key[2] over the same field, namely a description of a group gk and two commitment keys $[\mathbf{r}] \in \mathbb{G}^m$ $[\mathbf{s}] \in \mathbb{G}^{m-\ell+1}$.

We collectively denote with par, $[\mathbf{r}], [\mathbf{s}]$ this sets of parameters. The language of committed relaxed R1CS parametrized by these values is

$$\mathcal{L}_{\text{par},[\mathbf{r}],[\mathbf{s}]} = \big\{[e], u, [w], \mathbf{x} \mid \exists \mathbf{e}, \mathbf{w} \text{ s.t. } (1)\ [e] = [\mathbf{r}]^\top \mathbf{e},\ [w] = [\mathbf{s}]^\top \mathbf{w},\\ (2)\ \mathbf{Az} \circ \mathbf{Bz} = u \cdot \mathbf{Cz} + \mathbf{e}, (3)\ \mathbf{z} = (\mathbf{w}, \mathbf{x}, 1)\big\}$$

Essentially, this corresponds to the usual R1CS relation, but it has additional elements that allow folding instances.

We state the fact that there exists a non-interactive folding scheme for this language. We refer the reader to [KST21, Sec. 5] for the underlying details.

Theorem 39. *There exists a non-interactive 2-folding scheme for the family of languages of committed relaxed R1CS. The prover's computation is dominated by $O(m)$ field operations and the verifier's work is dominated by $O(\ell)$ field operations and a constant number of group and field operations. Both prover and verifier also need to perform a hash function computation.*

[2] We omit the requirement for a hiding key in our description.

This corresponds to [KST21, Construction 3] and is obtained by applying the Fiat-Shamir transform to the interactive folding scheme [KST21, Construction 2].

Efficiency. There is a minimal overhead for the prover, who -apart from a linear number of hash computation- does little more work than reading the witnesses. The verifier performs a logarithmic number of hash computations and group operations. It additionally needs to do $n \log M$ field operations, where M is the total number of folded statements. The latter part can be reduced to $\log M$ group operations if one considers a variation of the language where the part of the statement x is succinctly committed as well.

6.4 Applications

As we have discussed, selective verification can improve efficiency on applications with a single server serving multiple clients in a trustless way. It allows to amortize the server's costs across multiple queries from the clients, while only incuring a small overhead for the clients. We demonstrate this by considering the case of delegation of computation as a service and verifiable database delegation.

Delegation of computation as a service. For delegation of computation in a trustless setting, one would normally resort to some sort of SNARK, especially in cases where interaction is prohibitive. As we mentioned in the introduction, though, the proving costs for SNARKs are generally high. We demonstrate how to use folding schemes to mitigate this issue.

We will consider two cases: (1) each party needs to perform arbitrary computations and (2) all parties are interested in doing the same computation on different inputs. Especially in the latter case, we can greatly reduce the costs of the prover by means of folding schemes with selective verification.

Many SNARKs are constructed by separately considering some information theoretic part and a cryptographic primitive. Two main approaches are known: using interactive oracle proofs [BCS16] and vector commitments [CF13] and using algebraic [CHM+20] or polynomial [CFF+20] holographic proofs and polynomial commitments [KZG10]. In the former, the prover and verifier, after interacting, reduce the validity of the claim to the opening of some commitments to vectors at some random indices, while in the latter the validity of the statement is reduced to opening some polynomial commitments in random values. The interaction can be removed by means of the Fiat-Shamir transform.

In either case, we can use the folding constructions of the previous section to amortize the cost of the latter step: inner product arguments for the former and polynomial commitment for the latter[3]. Specifically, with each computational query, the prover performs the information

[3]In fact, both inner product arguments and polynomial commitment folding can be used for either approach but the presentation becomes more natural by using one approach for each.

theoretic part of the SNARK and refrains for the time from opening the vector or polynomial commitment. After multiple interactions with different verifiers, it folds all the (vector or polynomial) commitments to a single one, and opens the latter at some random indices or points respectively. The randomness can be derived by hashing the folded statement. Each individual verifier can now assert the folding proof as well as some evidence sent by the server asserting the inclusion of her statement.

If all parties are interested in performing the same computation on different inputs, one could use the NOVA approach. Specifically, the computation is encoded as a relaxed R1CS statement and the various instances of this statement are aggregated using the NOVA folding scheme compiled to support selective verification. As we discussed, a folding of this type of statements is very efficient. This is in construst to the previous case, since the SNARK information theoretic part (which needs to be in fact executed for each query to the proving server) is in fact costly for the prover. Considering the case of a single computation allows us to completely remove the need for this part and directly fold statements, which is not much costlier than simply reading the statements.

Verifiable Databases. In a verifiable database, a client outsources the storage of a database to a server in a trustless way. Specifically, the client only holds a small digest of the database and can query/modify the database in a verfiable way by means of communication with the prover. Such a construction can be built using vector commitments. The databases is encoded as a vector and the client only needs to hold the (constant size) commitment to the database. A query to the database can be done verifiable by asking the server to open the commitment to the desired locations. Furthermore, if the underlying commitment scheme is homomorphic (for example the Pedersen commitment), updating the database is efficient since one just needs to homomorphically update the digest by removing the old values and adding the new ones.

Consider the case where a server outsources storage to various clients. Naively implementing this would require that it sends an (expensive to produce) proof of opening for every query of every client to its database. Using a folding scheme with selective verification (for example the inner product language construction) can naturally minimize this cost.

In particular, each query to the server is answered without any verifiability guarantee; the clients simply get their responses and perform their updates acting in good faith. However, periodically, the server folds all the claims from all the clients using the folding scheme and publish a single statement and individualized proofs for each client to convince about the validity of all statements of one period. Due to the efficiency of the folding scheme, the amortized cost for this is much less than giving an individual proof for each claim.

Bibliography

[ABLZ17] Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, and Michal Zajac. "A Subversion-Resistant SNARK". In: *ASIACRYPT 2017, Part III*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Vol. 10626. LNCS. Springer, Heidelberg, Dec. 2017, pp. 3–33. DOI: 10.1007/978-3-319-70700-6_1.

[ACC+16] Prabhanjan Ananth, Yu-Chi Chen, Kai-Min Chung, Huijia Lin, and Wei-Kai Lin. "Delegating RAM Computations with Adaptive Soundness and Privacy". In: *TCC 2016-B, Part II*. Ed. by Martin Hirt and Adam D. Smith. Vol. 9986. LNCS. Springer, Heidelberg, Oct. 2016, pp. 3–30. DOI: 10.1007/978-3-662-53644-5_1.

[AFG+16] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. "Structure-Preserving Signatures and Commitments to Group Elements". In: *Journal of Cryptology* 29.2 (Apr. 2016), pp. 363–421. DOI: 10.1007/s00145-014-9196-7.

[AS92] Sanjeev Arora and Shmuel Safra. "Probabilistic Checking of Proofs; A New Characterization of NP". In: *33rd FOCS*. IEEE Computer Society Press, Oct. 1992, pp. 2–13. DOI: 10.1109/SFCS.1992.267824.

[Bab85] László Babai. "Trading Group Theory for Randomness". In: *17th ACM STOC*. ACM Press, May 1985, pp. 421–429. DOI: 10.1145/22145.22192.

[BB11] Dan Boneh and Xavier Boyen. "Efficient Selective Identity-Based Encryption Without Random Oracles". In: *Journal of Cryptology* 24.4 (Oct. 2011), pp. 659–693. DOI: 10.1007/s00145-010-9078-6.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. "Bulletproofs: Short Proofs for Confidential Transactions and More". In: *2018 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2018, pp. 315–334. DOI: 10.1109/SP.2018.00020.

[BBF19] Dan Boneh, Benedikt Bünz, and Ben Fisch. "Batching Techniques for Accumulators with Applications to IOPs and Stateless Blockchains". In: *CRYPTO 2019, Part I*. Ed. by Alexandra Boldyreva and Daniele Micciancio.

Vol. 11692. LNCS. Springer, Heidelberg, Aug. 2019, pp. 561–586. DOI: 10.1007/978-3-030-26948-7_20.

[BBHR19] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. "Scalable Zero Knowledge with No Trusted Setup". In: *CRYPTO 2019, Part III*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11694. LNCS. Springer, Heidelberg, Aug. 2019, pp. 701–732. DOI: 10.1007/978-3-030-26954-8_23.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. "Short Group Signatures". In: *CRYPTO 2004*. Ed. by Matthew Franklin. Vol. 3152. LNCS. Springer, Heidelberg, Aug. 2004, pp. 41–55. DOI: 10.1007/978-3-540-28628-8_3.

[BCC+16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. "Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting". In: *EUROCRYPT 2016, Part II*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. LNCS. Springer, Heidelberg, May 2016, pp. 327–357. DOI: 10.1007/978-3-662-49896-5_12.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. "Recursive composition and bootstrapping for SNARKS and proof-carrying data". In: *45th ACM STOC*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM Press, June 2013, pp. 111–120. DOI: 10.1145/2488608.2488623.

[BCG+14a] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. *Zerocash: Decentralized Anonymous Payments from Bitcoin*. Cryptology ePrint Archive, Report 2014/349. https://eprint.iacr.org/2014/349. 2014.

[BCG+14b] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. "Zerocash: Decentralized Anonymous Payments from Bitcoin". In: *2014 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2014, pp. 459–474. DOI: 10.1109/SP.2014.36.

[BCG20] Jonathan Bootle, Alessandro Chiesa, and Jens Groth. "Linear-Time Arguments with Sublinear Verification from Tensor Codes". In: *TCC 2020, Part II*. Ed. by Rafael Pass and Krzysztof Pietrzak. Vol. 12551. LNCS. Springer, Heidelberg, Nov. 2020, pp. 19–46. DOI: 10.1007/978-3-030-64378-2_2.

[BCHO22] Jonathan Bootle, Alessandro Chiesa, Yuncong Hu, and Michele Orrù. "Gemini: Elastic SNARKs for Diverse Environments". In: *IACR Cryptol. ePrint Arch.* (2022), p. 420. URL: https://eprint.iacr.org/2022/420.

[BCL+21] Benedikt Bünz, Alessandro Chiesa, William Lin, Pratyush Mishra, and Nicholas Spooner. "Proof-Carrying Data Without Succinct Arguments". In: *CRYPTO 2021, Part I*. Ed. by Tal Malkin and Chris Peikert. Vol. 12825. LNCS. Virtual Event: Springer, Heidelberg, Aug. 2021, pp. 681–710. DOI: 10.1007/978-3-030-84242-0_24.

[BCL20] Jonathan Bootle, Alessandro Chiesa, and Siqi Liu. "Zero-Knowledge Succinct Arguments with a Linear-Time Prover". In: *IACR Cryptol. ePrint Arch.* (2020), p. 1527. URL: https://eprint.iacr.org/2020/1527.

[BCMS20] Benedikt Bünz, Alessandro Chiesa, Pratyush Mishra, and Nicholas Spooner. "Recursive Proof Composition from Accumulation Schemes". In: *TCC 2020, Part II*. Ed. by Rafael Pass and Krzysztof Pietrzak. Vol. 12551. LNCS. Springer, Heidelberg, Nov. 2020, pp. 1–18. DOI: 10.1007/978-3-030-64378-2_1.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. "Interactive Oracle Proofs". In: *TCC 2016-B, Part II*. Ed. by Martin Hirt and Adam D. Smith. Vol. 9986. LNCS. Springer, Heidelberg, Oct. 2016, pp. 31–60. DOI: 10.1007/978-3-662-53644-5_2.

[BDFG21] Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. "Halo Infinite: Proof-Carrying Data from Additive Polynomial Commitments". In: *CRYPTO 2021, Part I*. Ed. by Tal Malkin and Chris Peikert. Vol. 12825. LNCS. Virtual Event: Springer, Heidelberg, Aug. 2021, pp. 649–680. DOI: 10.1007/978-3-030-84242-0_23.

[BDH+19] Michael Backes, Nico Döttling, Lucjan Hanzlik, Kamil Kluczniak, and Jonas Schneider. "Ring Signatures: Logarithmic-Size, No Setup - from Standard Assumptions". In: *EUROCRYPT 2019, Part III*. Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11478. LNCS. Springer, Heidelberg, May 2019, pp. 281–311. DOI: 10.1007/978-3-030-17659-4_10.

[BFL20] Balthazar Bauer, Georg Fuchsbauer, and Julian Loss. "A Classification of Computational Assumptions in the Algebraic Group Model". In: *CRYPTO 2020, Part II*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12171. LNCS. Springer, Heidelberg, Aug. 2020, pp. 121–151. DOI: 10.1007/978-3-030-56880-1_5.

[BFS16] Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. "NIZKs with an Untrusted CRS: Security in the Face of Parameter Subversion". In: *ASIACRYPT 2016, Part II*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10032. LNCS. Springer, Heidelberg, Dec. 2016, pp. 777–804. DOI: 10.1007/978-3-662-53890-6_26.

[BFS20] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. "Transparent SNARKs from DARK Compilers". In: *EUROCRYPT 2020, Part I*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12105. LNCS. Springer, Heidelberg, May 2020, pp. 677–706. DOI: 10.1007/978-3-030-45721-1_24.

[BGH19] Sean Bowe, Jack Grigg, and Daira Hopwood. *Halo: Recursive Proof Composition without a Trusted Setup*. Cryptology ePrint Archive, Report 2019/1021. https://eprint.iacr.org/2019/1021. 2019.

[BGL+15] Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. *Succinct Randomized Encodings and their Applications*. Cryptology ePrint Archive, Report 2015/356. https://eprint.iacr.org/2015/356. 2015.

[BGV11] Siavosh Benabbas, Rosario Gennaro, and Yevgeniy Vahlis. "Verifiable Delegation of Computation over Large Datasets". In: *CRYPTO 2011*. Ed. by Phillip Rogaway. Vol. 6841. LNCS. Springer, Heidelberg, Aug. 2011, pp. 111–131. DOI: 10.1007/978-3-642-22792-9_7.

[BHK17] Zvika Brakerski, Justin Holmgren, and Yael Tauman Kalai. "Non-interactive delegation and batch NP verification from standard computational assumptions". In: *49th ACM STOC*. Ed. by Hamed Hatami, Pierre McKenzie, and Valerie King. ACM Press, June 2017, pp. 474–482. DOI: 10.1145/3055399.3055497.

[BKK+18] Saikrishna Badrinarayanan, Yael Tauman Kalai, Dakshita Khurana, Amit Sahai, and Daniel Wichs. "Succinct delegation for low-space non-deterministic computation". In: *50th ACM STOC*. Ed. by Ilias Diakonikolas, David Kempe, and Monika Henzinger. ACM Press, June 2018, pp. 709–721. DOI: 10.1145/3188745.3188924.

[BL96] Dan Boneh and Richard J. Lipton. "Algorithms for Black-Box Fields and their Application to Cryptography (Extended Abstract)". In: *CRYPTO'96*. Ed. by Neal Koblitz. Vol. 1109. LNCS. Springer, Heidelberg, Aug. 1996, pp. 283–297. DOI: 10.1007/3-540-68697-5_22.

[BMM+21] Benedikt Bünz, Mary Maller, Pratyush Mishra, Nirvan Tyagi, and Psi Vesely. "Proofs for Inner Pairing Products and Applications". In: *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part III*. Ed. by Mehdi Tibouchi and Huaxiong Wang. Vol. 13092. Lecture Notes in Computer Science. Springer, 2021, pp. 65–97. DOI: 10.1007/978-3-030-92078-4_3. URL: https://doi.org/10.1007/978-3-030-92078-4%5C_3.

[BR93] Mihir Bellare and Phillip Rogaway. "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols". In: *ACM CCS 93*. Ed. by Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby. ACM Press, Nov. 1993, pp. 62–73. DOI: 10.1145/168588.168596.

[CCC+16] Yu-Chi Chen, Sherman S. M. Chow, Kai-Min Chung, Russell W. F. Lai, Wei-Kai Lin, and Hong-Sheng Zhou. "Cryptography for Parallel RAM from Indistinguishability Obfuscation". In: *ITCS 2016*. Ed. by Madhu Sudan. Jan. 2016, pp. 179–190. DOI: 10.1145/2840728.2840769.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. "Fiat-Shamir: from practice to theory". In: *51st ACM STOC*. Ed. by Moses Charikar and Edith Cohen. ACM Press, June 2019, pp. 1082–1090. DOI: 10.1145/3313276.3316380.

[CF13] Dario Catalano and Dario Fiore. "Vector Commitments and Their Applications". In: *PKC 2013*. Ed. by Kaoru Kurosawa and Goichiro Hanaoka. Vol. 7778. LNCS. Springer, Heidelberg, Feb. 2013, pp. 55–72. DOI: 10.1007/978-3-642-36362-7_5.

[CFF+20] Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. *Lunar: a Toolbox for More Efficient Universal and Updatable zkSNARKs and Commit-and-Prove Extensions*. Cryptology ePrint Archive, Report 2020/1069. https://eprint.iacr.org/2020/1069. 2020.

[CFG+20] Matteo Campanelli, Dario Fiore, Nicola Greco, Dimitris Kolonelos, and Luca Nizzardo. "Incrementally Aggregatable Vector Commitments and Applications to Verifiable Decentralized Storage". In: *ASIACRYPT 2020, Part II*. Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12492. LNCS. Springer, Heidelberg, Dec. 2020, pp. 3–35. DOI: 10.1007/978-3-030-64834-3_1.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. "The Random Oracle Methodology, Revisited". In: *J. ACM* 51.4 (July 2004), pp. 557–594. ISSN: 0004-5411. DOI: 10.1145/1008731.1008734. URL: https://doi.org/10.1145/1008731.1008734.

[CHJV15] Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. "Succinct Garbling and Indistinguishability Obfuscation for RAM Programs". In: *47th ACM STOC*. Ed. by Rocco A. Servedio and Ronitt Rubinfeld. ACM Press, June 2015, pp. 429–437. DOI: 10.1145/2746539.2746621.

[CHM+20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas P. Ward. "Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS". In: *EUROCRYPT 2020, Part I*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12105. LNCS. Springer, Heidelberg, May 2020, pp. 738–768. DOI: 10.1007/978-3-030-45721-1_26.

[CPZ18] Alexander Chepurnoy, Charalampos Papamanthou, and Yupeng Zhang. *Edrax: A Cryptocurrency with Stateless Transaction Validation*. Cryptology ePrint Archive, Report 2018/968. https://eprint.iacr.org/2018/968. 2018.

[CS02] Ronald Cramer and Victor Shoup. "Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption". In: *EUROCRYPT 2002*. Ed. by Lars R. Knudsen. Vol. 2332. LNCS. Springer, Heidelberg, Apr. 2002, pp. 45–64. DOI: 10.1007/3-540-46035-7_4.

[Dam92] Ivan Damgård. "Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks". In: *CRYPTO'91*. Ed. by Joan Feigenbaum. Vol. 576. LNCS. Springer, Heidelberg, Aug. 1992, pp. 445–456. DOI: 10.1007/3-540-46766-1_36.

[DFGK14] George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. "Square Span Programs with Applications to Succinct NIZK Arguments". In: *ASIACRYPT 2014, Part I*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8873. LNCS. Springer, Heidelberg, Dec. 2014, pp. 532–550. DOI: 10.1007/978-3-662-45611-8_28.

[DGP+19] Vanesa Daza, Alonso González, Zaira Pindado, Carla Ràfols, and Javier Silva. "Shorter Quadratic QA-NIZK Proofs". In: *PKC 2019, Part I*. Ed. by Dongdai Lin and Kazue Sako. Vol. 11442. LNCS. Springer, Heidelberg, Apr. 2019, pp. 314–343. DOI: 10.1007/978-3-030-17253-4_11.

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. "An Algebraic Framework for Diffie-Hellman Assumptions". In: *CRYPTO 2013, Part II*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8043. LNCS. Springer, Heidelberg, Aug. 2013, pp. 129–147. DOI: 10.1007/978-3-642-40084-1_8.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. "The Algebraic Group Model and its Applications". In: *CRYPTO 2018, Part II*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10992. LNCS. Springer, Heidelberg, Aug. 2018, pp. 33–62. DOI: 10.1007/978-3-319-96881-0_2.

[FLPS20] Prastudy Fauzi, Helger Lipmaa, Zaira Pindado, and Janno Siim. *Somewhere Statistically Binding Commitment Schemes with Applications*. Cryptology ePrint Archive, Report 2020/652. https://eprint.iacr.org/2020/652. 2020.

[FS87] Amos Fiat and Adi Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems". In: *CRYPTO'86*. Ed. by Andrew M. Odlyzko. Vol. 263. LNCS. Springer, Heidelberg, Aug. 1987, pp. 186–194. DOI: 10.1007/3-540-47721-7_12.

[Fuc18] Georg Fuchsbauer. "Subversion-Zero-Knowledge SNARKs". In: *PKC 2018, Part I*. Ed. by Michel Abdalla and Ricardo Dahab. Vol. 10769. LNCS. Springer, Heidelberg, Mar. 2018, pp. 315–347. DOI: 10.1007/978-3-319-76578-5_11.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. "Quadratic Span Programs and Succinct NIZKs without PCPs". In: *EUROCRYPT 2013*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. LNCS. Springer, Heidelberg, May 2013, pp. 626–645. DOI: 10.1007/978-3-642-38348-9_37.

[GHR15] Alonso González, Alejandro Hevia, and Carla Ràfols. "QA-NIZK Arguments in Asymmetric Groups: New Tools and New Constructions". In: *ASIACRYPT 2015, Part I*. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9452. LNCS. Springer, Heidelberg, Nov. 2015, pp. 605–629. DOI: 10.1007/978-3-662-48797-6_25.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. "On the (In)security of the Fiat-Shamir Paradigm". In: *44th FOCS*. IEEE Computer Society Press, Oct. 2003, pp. 102–115. DOI: 10.1109/SFCS.2003.1238185.

[GK16] Shafi Goldwasser and Yael Tauman Kalai. "Cryptographic Assumptions: A Position Paper". In: *TCC 2016-A, Part I*. Ed. by Eyal Kushilevitz and Tal Malkin. Vol. 9562. LNCS. Springer, Heidelberg, Jan. 2016, pp. 505–522. DOI: 10.1007/978-3-662-49096-9_21.

[GKM+18] Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. "Updatable and Universal Common Reference Strings with Applications to zk-SNARKs". In: *CRYPTO 2018, Part III*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10993. LNCS. Springer, Heidelberg, Aug. 2018, pp. 698–728. DOI: 10.1007/978-3-319-96878-0_24.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. "Delegating computation: interactive proofs for muggles". In: *40th ACM STOC*. Ed. by Richard E. Ladner and Cynthia Dwork. ACM Press, May 2008, pp. 113–122. DOI: 10.1145/1374376.1374396.

[GM17] Jens Groth and Mary Maller. "Snarky Signatures: Minimal Signatures of Knowledge from Simulation-Extractable SNARKs". In: *CRYPTO 2017, Part II*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10402. LNCS. Springer, Heidelberg, Aug. 2017, pp. 581–612. DOI: 10.1007/978-3-319-63715-0_20.

[GMN21] Nicolas Gailly, Mary Maller, and Anca Nitulescu. *SnarkPack: Practical SNARK Aggregation*. Cryptology ePrint Archive, Report 2021/529. https://eprint.iacr.org/2021/529. 2021.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract)". In: *17th ACM STOC*. ACM Press, May 1985, pp. 291–304. DOI: 10.1145/22145.22178.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. "How to Prove all NP-Statements in Zero-Knowledge, and a Methodology of Cryptographic Protocol Design". In: *CRYPTO'86*. Ed. by Andrew M. Odlyzko. Vol. 263. LNCS. Springer, Heidelberg, Aug. 1987, pp. 171–185. DOI: 10.1007/3-540-47721-7_11.

[Gol11] O. Goldreich. *Stories about Shimon Even (by Oded)*. www.wisdom.weizmann.ac.il/~oded/even-stories.html. Accessed: 2021-07-03. 2011.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. "Perfect Non-interactive Zero Knowledge for NP". In: *EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. LNCS. Springer, Heidelberg, May 2006, pp. 339–358. DOI: 10.1007/11761679_21.

[GR16] Alonso González and Carla Ràfols. "New Techniques for Non-interactive Shuffle and Range Arguments". In: *ACNS 16*. Ed. by Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider. Vol. 9696. LNCS. Springer, Heidelberg, June 2016, pp. 427–444. DOI: 10.1007/978-3-319-39555-5_23.

[GR19] Alonso González and Carla Ràfols. "Shorter Pairing-Based Arguments Under Standard Assumptions". In: *ASIACRYPT 2019, Part III*. Ed. by Steven D. Galbraith and Shiho Moriai. Vol. 11923. LNCS. Springer, Heidelberg, Dec. 2019, pp. 728–757. DOI: 10.1007/978-3-030-34618-8_25.

[Gro10] Jens Groth. "Short Pairing-Based Non-interactive Zero-Knowledge Arguments". In: *ASIACRYPT 2010*. Ed. by Masayuki Abe. Vol. 6477. LNCS. Springer, Heidelberg, Dec. 2010, pp. 321–340. DOI: 10.1007/978-3-642-17373-8_19.

[Gro16] Jens Groth. "On the Size of Pairing-Based Non-interactive Arguments". In: *EUROCRYPT 2016, Part II*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. LNCS. Springer, Heidelberg, May 2016, pp. 305–326. DOI: 10.1007/978-3-662-49896-5_11.

[GRWZ20] Sergey Gorbunov, Leonid Reyzin, Hoeteck Wee, and Zhenfei Zhang. "Pointproofs: Aggregating Proofs for Multiple Vector Commitments". In: *ACM CCS 2020*. Ed. by Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna. ACM Press, Nov. 2020, pp. 2007–2023. DOI: 10.1145/3372297.3417244.

[GS08] Jens Groth and Amit Sahai. "Efficient Non-interactive Proof Systems for Bilinear Groups". In: *EUROCRYPT 2008*. Ed. by Nigel P. Smart. Vol. 4965. LNCS. Springer, Heidelberg, Apr. 2008, pp. 415–432. DOI: 10.1007/978-3-540-78967-3_24.

[GW11] Craig Gentry and Daniel Wichs. "Separating succinct non-interactive arguments from all falsifiable assumptions". In: *43rd ACM STOC*. Ed. by Lance Fortnow and Salil P. Vadhan. ACM Press, June 2011, pp. 99–108. DOI: 10.1145/1993636.1993651.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. *PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge*. Cryptology ePrint Archive, Report 2019/953. https://eprint.iacr.org/2019/953. 2019.

[HW15] Pavel Hubacek and Daniel Wichs. "On the Communication Complexity of Secure Function Evaluation with Long Output". In: *ITCS 2015*. Ed. by Tim Roughgarden. Jan. 2015, pp. 163–172. DOI: 10.1145/2688073.2688105.

[JKKZ20] Ruta Jawale, Yael Tauman Kalai, Dakshita Khurana, and Rachel Zhang. *SNARGs for Bounded Depth Computations and PPAD Hardness from Sub-Exponential LWE*. Cryptology ePrint Archive, Report 2020/980. https://eprint.iacr.org/2020/980. 2020.

[JR13] Charanjit S. Jutla and Arnab Roy. "Shorter Quasi-Adaptive NIZK Proofs for Linear Subspaces". In: *ASIACRYPT 2013, Part I*. Ed. by Kazue Sako and Palash Sarkar. Vol. 8269. LNCS. Springer, Heidelberg, Dec. 2013, pp. 1–20. DOI: 10.1007/978-3-642-42033-7_1.

[JR14] Charanjit S. Jutla and Arnab Roy. "Switching Lemma for Bilinear Tests and Constant-Size NIZK Proofs for Linear Subspaces". In: *CRYPTO 2014, Part II*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8617. LNCS. Springer, Heidelberg, Aug. 2014, pp. 295–312. DOI: 10.1007/978-3-662-44381-1_17.

[Kil92] Joe Kilian. "A Note on Efficient Zero-Knowledge Proofs and Arguments (Extended Abstract)". In: *24th ACM STOC*. ACM Press, May 1992, pp. 723–732. DOI: 10.1145/129712.129782.

[KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. "Indistinguishability Obfuscation for Turing Machines with Unbounded Memory". In: *47th ACM STOC*. Ed. by Rocco A. Servedio and Ronitt Rubinfeld. ACM Press, June 2015, pp. 419–428. DOI: 10.1145/2746539.2746614.

[KNYY19] Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. "Exploring Constructions of Compact NIZKs from Various Assumptions". In: *CRYPTO 2019, Part III*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11694. LNCS. Springer, Heidelberg, Aug. 2019, pp. 639–669. DOI: 10.1007/978-3-030-26954-8_21.

[KNYY20] Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. "Compact NIZKs from Standard Assumptions on Bilinear Maps". In: *EUROCRYPT 2020, Part III*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12107. LNCS. Springer, Heidelberg, May 2020, pp. 379–409. DOI: 10.1007/978-3-030-45727-3_13.

[KP16] Yael Tauman Kalai and Omer Paneth. "Delegating RAM Computations". In: *TCC 2016-B, Part II*. Ed. by Martin Hirt and Adam D. Smith. Vol. 9986. LNCS. Springer, Heidelberg, Oct. 2016, pp. 91–118. DOI: 10.1007/978-3-662-53644-5_4.

[KPY18] Yael Kalai, Omer Paneth, and Lisa Yang. *On Publicly Verifiable Delegation From Standard Assumptions*. Cryptology ePrint Archive, Report 2018/776. https://eprint.iacr.org/2018/776. 2018.

[KPY19] Yael Tauman Kalai, Omer Paneth, and Lisa Yang. "How to delegate computations publicly". In: *51st ACM STOC*. Ed. by Moses Charikar and Edith Cohen. ACM Press, June 2019, pp. 1115–1124. DOI: 10.1145/3313276.3316411.

[KRR13] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. "Delegation for bounded space". In: *45th ACM STOC*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM Press, June 2013, pp. 565–574. DOI: 10.1145/2488608.2488679.

[KRR14] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. "How to delegate computations: the power of no-signaling proofs". In: *46th ACM STOC*. Ed. by David B. Shmoys. ACM Press, May 2014, pp. 485–494. DOI: 10.1145/2591796.2591809.

[KST21] Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. *Nova: Recursive Zero-Knowledge Arguments from Folding Schemes*. Cryptology ePrint Archive, Report 2021/370. https://eprint.iacr.org/2021/370. 2021.

[Kus18] John Kuszmaul. *Verkle trees*. 2018. URL: https://math.mit.edu/research/highschool/primes/materials/2018/Kuszmaul.pdf.

[KW15] Eike Kiltz and Hoeteck Wee. "Quasi-Adaptive NIZK for Linear Subspaces Revisited". In: *EUROCRYPT 2015, Part II*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9057. LNCS. Springer, Heidelberg, Apr. 2015, pp. 101–128. DOI: 10.1007/978-3-662-46803-6_4.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. "Constant-Size Commitments to Polynomials and Their Applications". In: *ASIACRYPT 2010*. Ed. by Masayuki Abe. Vol. 6477. LNCS. Springer, Heidelberg, Dec. 2010, pp. 177–194. DOI: 10.1007/978-3-642-17373-8_11.

[LM19] Russell W. F. Lai and Giulio Malavolta. "Subvector Commitments with Application to Succinct Arguments". In: *CRYPTO 2019, Part I*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11692. LNCS. Springer, Heidelberg, Aug. 2019, pp. 530–560. DOI: 10.1007/978-3-030-26948-7_19.

[LPJY13] Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. "Linearly Homomorphic Structure-Preserving Signatures and Their Applications". In: *CRYPTO 2013, Part II*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8043. LNCS. Springer, Heidelberg, Aug. 2013, pp. 289–307. DOI: 10.1007/978-3-642-40084-1_17.

[LRY16] Benoît Libert, Somindu C. Ramanna, and Moti Yung. "Functional Commitment Schemes: From Polynomial Commitments to Pairing-Based Accumulators from Simple Assumptions". In: *ICALP 2016*. Ed. by Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi. Vol. 55. LIPIcs. Schloss Dagstuhl, July 2016, 30:1–30:14. DOI: 10.4230/LIPIcs.ICALP.2016.30.

[LY10] Benoît Libert and Moti Yung. "Concise Mercurial Vector Commitments and Independent Zero-Knowledge Sets with Short Proofs". In: *TCC 2010*. Ed. by Daniele Micciancio. Vol. 5978. LNCS. Springer, Heidelberg, Feb. 2010, pp. 499–517. DOI: 10.1007/978-3-642-11799-2_30.

[MBKM19] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. "Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updatable Structured Reference Strings". In: *ACM CCS 2019*. Ed. by Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz. ACM Press, Nov. 2019, pp. 2111–2128. DOI: 10.1145/3319535.3339817.

[Mic94] Silvio Micali. "CS Proofs (Extended Abstracts)". In: *35th FOCS*. IEEE Computer Society Press, Nov. 1994, pp. 436–453. DOI: 10.1109/SFCS.1994.365746.

[MRV16] Paz Morillo, Carla Ràfols, and Jorge Luis Villar. "The Kernel Matrix Diffie-Hellman Assumption". In: *ASIACRYPT 2016, Part I*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10031. LNCS. Springer, Heidelberg, Dec. 2016, pp. 729–758. DOI: 10.1007/978-3-662-53887-6_27.

[Nao03] Moni Naor. "On Cryptographic Assumptions and Challenges (Invited Talk)". In: *CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. LNCS. Springer, Heidelberg, Aug. 2003, pp. 96–109. DOI: 10.1007/978-3-540-45146-4_6.

[OPWW15] Tatsuaki Okamoto, Krzysztof Pietrzak, Brent Waters, and Daniel Wichs. "New Realizations of Somewhere Statistically Binding Hashing and Positional Accumulators". In: *ASIACRYPT 2015, Part I*. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9452. LNCS. Springer, Heidelberg, Nov. 2015, pp. 121–145. DOI: 10.1007/978-3-662-48797-6_6.

[PR17] Omer Paneth and Guy N. Rothblum. "On Zero-Testable Homomorphic Encryption and Publicly Verifiable Non-interactive Arguments". In: *TCC 2017, Part II*. Ed. by Yael Kalai and Leonid Reyzin. Vol. 10678. LNCS. Springer, Heidelberg, Nov. 2017, pp. 283–315. DOI: 10.1007/978-3-319-70503-3_9.

[PST13] Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. "Signatures of Correct Computation". In: *TCC 2013*. Ed. by Amit Sahai. Vol. 7785. LNCS. Springer, Heidelberg, Mar. 2013, pp. 222–242. DOI: 10.1007/978-3-642-36594-2_13.

[RR21] Noga Ron-Zewi and Ron Rothblum. "Proving as Fast as Computing: Succinct Arguments with Constant Prover Overhead". In: *Electron. Colloquium Comput. Complex.* (2021), p. 180. URL: https://eccc.weizmann.ac.il/report/2021/180.

[RRR16] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. "Constant-round interactive proofs for delegating computation". In: *48th ACM STOC*. Ed. by Daniel Wichs and Yishay Mansour. ACM Press, June 2016, pp. 49–62. DOI: 10.1145/2897518.2897652.

[RS20] Carla Ràfols and Javier Silva. *QA-NIZK Arguments of Same Opening for Bilateral Commitments*. Cryptology ePrint Archive, Report 2020/569. https://eprint.iacr.org/2020/569. 2020.

[RZ21] Carla Ràfols and Arantxa Zapico. "An Algebraic Framework for Universal and Updatable SNARKs". In: *CRYPTO 2021, Part I*. Ed. by Tal Malkin and Chris Peikert. Vol. 12825. LNCS. Virtual Event: Springer, Heidelberg, Aug. 2021, pp. 774–804. DOI: 10.1007/978-3-030-84242-0_27.

[SCP+22] Shravan Srinivasan, Alexander Chepurnoy, Charalampos Papamanthou, Alin Tomescu, and Yupeng Zhang. "Hyperproofs: Aggregating and Maintaining Proofs in Vector Commitments". In: *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022. URL: https://www.usenix.org/conference/usenixsecurity22/presentation/srinivasan.

[Sho97] Victor Shoup. "Lower Bounds for Discrete Logarithms and Related Problems". In: *EUROCRYPT'97*. Ed. by Walter Fumy. Vol. 1233. LNCS. Springer, Heidelberg, May 1997, pp. 256–266. DOI: 10.1007/3-540-69053-0_18.

[Val08] Paul Valiant. "Incrementally Verifiable Computation or Proofs of Knowledge Imply Time/Space Efficiency". In: *TCC 2008*. Ed. by Ran Canetti. Vol. 4948. LNCS. Springer, Heidelberg, Mar. 2008, pp. 1–18. DOI: 10.1007/978-3-540-78524-8_1.

[Vil12] Jorge Luis Villar. "Optimal Reductions of Some Decisional Problems to the Rank Problem". In: *ASIACRYPT 2012*. Ed. by Xiaoyun Wang and Kazue Sako. Vol. 7658. LNCS. Springer, Heidelberg, Dec. 2012, pp. 80–97. DOI: 10.1007/978-3-642-34961-4_7.

[WTs+18] Riad S. Wahby, Ioanna Tzialla, abhi shelat, Justin Thaler, and Michael Walfish. "Doubly-Efficient zkSNARKs Without Trusted Setup". In: *2018 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2018, pp. 926–943. DOI: 10.1109/SP.2018.00060.

[XZZ+19] Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. "Libra: Succinct Zero-Knowledge Proofs with Optimal Prover Computation". In: *CRYPTO 2019, Part III*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11694. LNCS. Springer, Heidelberg, Aug. 2019, pp. 733–764. DOI: 10.1007/978-3-030-26954-8_24.

[YLF+21] Thomas Yurek, Licheng Luo, Jaiden Fairoze, Aniket Kate, and Andrew Miller. *hbACSS: How to Robustly Share Many Secrets*. Cryptology ePrint Archive, Report 2021/159. https://eprint.iacr.org/2021/159. 2021.

[ZCa21] ZCash. *Parameter Generation for the ZCash cryptocurrency*. https://z.cash/technology/paramgen/. Accessed: 2021-07-9. 2021.

[ZGK+17a] Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. *A Zero-Knowledge Version of vSQL*. Cryptology ePrint Archive, Report 2017/1146. https://eprint.iacr.org/2017/1146. 2017.

[ZGK+17b] Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. "vSQL: Verifying Arbitrary SQL Queries over Dynamic Outsourced Databases". In: *2017 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2017, pp. 863–880. DOI: 10.1109/SP.2017.43.

List of Publications

Conference Proceedings

[DHS+22] Vanesa Daza, Abida Haque, Alessandra Scafuro, Alexandros Zacharakis, and Arantxa Zapico. "Mutual Accountability Layer: Accountable Anonymity Within Accountable Trust". In: *Cyber Security, Cryptology, and Machine Learning - 6th International Symposium, CSCML 2022, Be'er Sheva, Israel, June 30 - July 1, 2022, Proceedings*. Ed. by Shlomi Dolev, Jonathan Katz, and Amnon Meisels. Vol. 13301. Lecture Notes in Computer Science. Springer, 2022, pp. 318–336. DOI: 10.1007/978-3-031-07689-3_24. URL: https://doi.org/10.1007/978-3-031-07689-3%5C_24.

[DRZ20] Vanesa Daza, Carla Ràfols, and Alexandros Zacharakis. "Updateable Inner Product Argument with Logarithmic Verifier and Applications". In: *Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part I*. Ed. by Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas. Vol. 12110. Lecture Notes in Computer Science. Springer, 2020, pp. 527–557. DOI: 10.1007/978-3-030-45374-9_18. URL: https://doi.org/10.1007/978-3-030-45374-9%5C_18.

[GZ21] Alonso González and Alexandros Zacharakis. "Fully-Succinct Publicly Verifiable Delegation from Constant-Size Assumptions". In: *Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part I*. Ed. by Kobbi Nissim and Brent Waters. Vol. 13042. Lecture Notes in Computer Science. Springer, 2021, pp. 529–557. DOI: 10.1007/978-3-030-90459-3_18. URL: https://doi.org/10.1007/978-3-030-90459-3%5C_18.

Pre-prints

[CNR+22] Matteo Campanelli, Anca Nitulescu, Carla Ràfols, Alexandros Zacharakis, and Arantxa Zapico. "Linear-map Vector Commitments and their Practical Applications". In: *IACR Cryptol. ePrint Arch.* (2022), p. 705. URL: https://eprint.iacr.org/2022/705.