THÈSE DE DOCTORAT DE
l'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Julien KEUFFER

Pour obtenir le grade de
DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse :

Calcul vérifiable et vérification biométrique.

soutenue le 25 février 2019,

devant le jury composé de :

M. Josep Domingo-Ferrer	Examinateur
M. Dario Fiore	Examinateur
M. Sébastien Gambs	Rapporteur
Mme. Aikaterini Mitrokotsa	Rapporteuse
M. Refik Molva	Directeur de thèse
M. Damien Vergnaud	Examinateur

Contents

Chapter 1

Introduction

Since the early 2000s, cloud computing has been a predominant paradigm for the entire computing industry offering significant advantages for customers. The pay for use model enables for instance individuals or small and medium companies to benefit from previously unaffordable computing resources. Thanks to the elasticity of cloud computing resources, companies can enjoy from extensible computing and storage capacity without having to pay for the cost of ownership for software and hardware. The advantages of cloud computing do however come with the cost of new vulnerabilities due to the lack of guarantees in the setting of outsourced computations, such as the guarantee for correctness: there is no reason to trust a third party to which a computational task has been delegated and running a computation by operating a network of computers is complex. Moreover, unintended errors can happen, for instance due to hardware failure, fire, earthquakes or lightning strikes. Besides, existing cloud computing companies deny any liability in case of errors. This trust issue raises the need to *verify* a computation performed by a third party cloud service provider.

A tentative solution to verify a computation is to leverage replication: the same computation is executed on several hosts and a decision is taken based on the comparison of the results produced by each host. For instance the distributed computing platform SETI@home [ACK+02] shares several replicas of the same fast Fourier computation among clients and verifies the returned results by means of a majority rule. Nonetheless, this majority rule does not guarantee that the result is correct. A further possibility is to rely on trusted hardware which ensures that the running code has not been tampered with but again, the correctness of the result is not guaranteed. Moreover, the size of the code that can be embedded in the trusted hardware is limited.

An alternative approach leverages recent results on the notion of proof that are achieved in the field of cryptography and complexity theory. The seminal work of Goldwasser et al. [GMR85] proposed a new way to envision the notion of proof and designed a protocol where, thanks to randomness and interactions between a prover and a verifier, the validity of a proof could be established. With an adequate language, the proof can thus guarantee the correct execution of a program. Regarding delegation of computations, the verification by means of a protocol driven by a trusted computer had already been mentioned in 1991 as a potential application in a seminal article by Babai et al. [BFLS91]:

> "In this setup, a single reliable PC can monitor the operation of a herd of supercomputers working with possibly extremely powerful but unreliable

software and untested hardware."

Checking computation in polylogarithmic time – Babai et al.

Babai et al.'s article belonged to a series of articles that led to the celebrated Probabilistically Checkable Proofs (PCP) theorem [AS98]. This seminal result deals with probabilistic proofs and states that for every language in **NP** and for every instance that belongs to the language, it is possible to produce a proof of membership that has to be checked in a constant number of locations to decide if the proof is correct. The decision holds with a certain probability that can be increased if more locations are consulted. Even if the PCP theorem theoretically solves the problem of verifying computations, a practical implementation of the scheme would have been impossible either because of the proof size or because of the time required to produce such a proof.

Furthermore, the advent of cloud computing drew back the attention of the computer science community to the question of verifying outsourced computations and the establishment of a new research field known as verifiable computation (VC). The goal of VC schemes is to attach a proof of correctness to the result of a computation. For that notion to be of practical interest, verifying the proof should additionally be more efficient than running the computation. Verifiable computing schemes offer therefore the possibility to delegate computations to an untrusted entity. At that time, the existing practical-oriented VC schemes leveraged replication of computations [ACK+02, KSC09, CRR11], relied on trusted hardware [SSW10] or were specialized for a particular operation [Fre77, BGV11]. Nonetheless, these assumptions were not relevant in the cloud setting because:

- the hardware in the cloud is often homogeneous: failures are thus correlated and several instances of the same computation can be simultaneously erroneous,
- it seems unrealistic to assume that each machine running in the cloud has a trusted hardware,
- the diversity of computations to verify ruled out specialized schemes.

Building on theoretical results published after the PCP theorem, a couple of research teams studied verifiable computation with implementation as explicit goal. Further to Goldwasser et al.'s seminal "Interactive Proofs for the Muggles" [GKR08], Cormode et al. [CMT12] and later Thaler [Tha13] came up with an efficient interactive proof system and implemented it. A different approach was taken by Setty et al. : inspired by Ishai et al.'s linear PCP [IKO07], they proposed and implemented an interactive argument [SMBW12, SVP+12]. Another important step toward practicality was the publication of Parno et al.'s Pinocchio [PHGR13], which leverages a new encoding of computations that could be efficiently embedded into a verifiable computation protocol. The encoding was made by means of quadratic arithmetic programs (QAP), as defined in Gennaro et al.'s groundbreaking article [GGPR13]. Parno et al. [PHGR13] also released a complete compilation toolchain that takes as input a program written in a subset of C and outputs a verifiable executable. However, Pinocchio's implementation was not completely open sourced, due to the use of a proprietary library. Due to their remarkably efficient encoding, QAPs were the baseline for several subsequent works: Ben-Sasson et al. optimized Parno et al.'s protocol [BCG+13, BCTV14b] and extended the expressiveness of the original scheme while Setty et al. [SBV+13] integrated QAPs

in their interactive argument scheme. Both research teams released the source code of their implementations [lib, pep].

It is important to note that for every general-purpose VC scheme, the program to be verified should be expressed as a circuit, namely an oriented acyclic graph whose nodes are operations, such as addition or multiplication, and whose edges, called *wires*, carry values of a finite field. Executing a program comes down to assigning the input values of the program to the nodes of in-degree 0 of the circuit, the result of the program being the output of the circuit. Theoretical results guarantee that every Turing machine can be represented as a circuit at the expense of a polynomial blow-up. Nonetheless, representing efficiently a program as a circuit is itself an active research area. There are two kinds of circuits: in *boolean circuits*, the finite field is $\mathbb{F}_2$ and the nodes are AND or XOR operations while in *arithmetic circuits*, the finite field is $\mathbb{F}_p, p > 2$ and the nodes are addition and multiplication in the finite field $\mathbb{F}_p$.

Three major approaches have been considered to produce proofs of computation correctness:

- *Interactive proofs*: allowing interactions between the prover and the verifier without any assumption on the computing power of a malicious prover,
- *Interactive arguments*: allowing interactions but designing systems that are secure against computationally bounded provers,
- *Non-interactive arguments*: requiring that the prover should produce the proof without interacting with the verifier.

Non-interactive proof systems are of particular interest since they enable the prover to execute the computation and to prove its correctness without being synchronized with the verifier. At a high-level view, a VC scheme is secure if it satisfies the following properties:

- *Correctness*: a verifier should accept every proof that has been computed by an honest prover.
- *Soundness*: no cheating prover can produce the proof of a false statement that will be accepted (except with negligible probability).
- *Efficiency*: verifying the proof should be more efficient than running the computation.

Some constructions reach additional interesting properties such as zero-knowledge: the prover can supply inputs in the computation to verify for which the proof will leak no information. For instance VC protocols based on QAPs enable to get zero-knowledge arguments with little extra effort for the prover. Moreover, these arguments also achieve the proof of knowledge property, in that if a prover convinces a verifier about the witness of a statement, he must be in possession of this witness. The resulting zero-knowledge arguments are called zero-knowledge succinct arguments of knowledge (zk-SNARKs) [BCCT12], for which many applications have been proposed, such as a crypto-currency [BCG+14] or a verifiable MapReduce function [BFR+13b, CFH+15] to cite a few.

The difficult trade-off between efficiency and expressiveness. The numerous works of the last years have brought verifiable computation closer to a practical use, however the overhead costs for the prover remains high [WB15]. One the one hand, protocols that come from interactive proofs are efficient but their expressiveness is limited to structured circuits. Hence, they do not reach full generality. On the other hand, the most expressive protocols, such as Ben-Sasson et al.'s TINYRAM [BCG+13] that reproduces a set of assembly instructions, are inefficient and cannot address practical scenarios. Protocols that build on QAP, whether building on interactive arguments [WSR+15] or on non-interactive arguments [PHGR13, CFH+15] reach a trade-off between expressiveness and efficiency. However, they are still too inefficient to deal with large computations.

Even if existing VC systems do not achieve the efficiency as required in practice (notably when comparing the verification of the computation to its *native* implementation [WB15, WJB+17a]), there are settings where the efficiency requirement is not relevant. When the VC scheme enables the prover to provide private inputs for a computation, efficiency makes no longer sense because the complete computation could not have been run by the verifier. Several state-of-art VC schemes ensure that the prover's private inputs do not leak in the proof by achieving a zero-knowledge property. QAP-based VC schemes notably require few extra work for the prover to get a zero-knowledge proof, resulting in zk-SNARK schemes. While the first efficient zero-knowledge proofs were designed for specialized computations that have a strong algebraic structure, the relative efficiency (at least compared to the generic transformation proposed in [GMW86]) of zk-SNARKs enables to compute zero-knowledge proofs for unstructured computations and furthermore to instantiate and enrich existing primitives.
In conclusion, all existing VC schemes adopt a certain trade-off between efficiency and expressiveness but none of them meets the requirements for an ideal VC scheme that should:

- be capable to deal with a large class of computations,
- have minimal proving costs, not only asymptotically but also in practice,
- have a short and efficiently verifiable proof,
- fulfill the efficiency requirement, namely verify the proof faster than executing the computation.

Moreover, there are situations where a VC scheme is required but the prover additionally wants to hide some of his inputs from the verifier. Hence, such VC schemes should also:

- enable the prover to provide private inputs (non-deterministic computations),
- provide additional properties like zero-knowledge or proof of knowledge,
- be able to produce short proofs that can be integrated in larger protocols.

Contributions

Our first contribution intends to shift the trade-off of non-interactive schemes by increasing efficiency. It starts from an expressive VC scheme and improves the efficiency of the latter by leveraging proof composition with an efficient VC scheme. The resulting scheme keeps its expressiveness while being more efficient each time it performs a computation for which an efficient and composable scheme exists. The second contribution leverages existing zk-SNARK scheme to build a protocol that enables to redact documents while keeping both integrity for the redacted document and privacy for the redacted information. These goals can be achieved thanks to the privacy protection in zk-SNARK that allows the prover to supply private inputs to the computation without revealing them to the verifier. Also leveraging the features of zk-SNARK, the third contribution is an authentication scheme where the entire authentication process is handled by an untrusted client device that provides the authentication result as a zk-SNARK proof.

Embedded Proofs [KMC18]

We present a new VC scheme tailored to efficiently deal with diverse type of operations, some of them requiring an expressive VC scheme (that we will call here "complex operations") while some others require a huge amount of computations (that we will call here "large operations"). To address this problematic, we propose to compose proof systems by embedding efficient but specialized VC schemes that can deal with large operations into a general-purpose VC scheme that is expressive enough to deal with complex operations. The embedding is based on the following idea: we compute the large operations outside the general-purpose proof system and leverage the specialized proof system to produce proofs of correctness. Then, we pass the result of these sub-computations along with their proofs to the general-purpose system. The latter embeds the verification algorithm of the specialized VC schemes and therefore checks that the result of the sub-computations is correct. The general-purpose VC scheme (GVC) also computes the complex operations. The final proof produced by the GVC scheme proves that all the sub-computations are correct (since the verification of their proof has passed) and that the complex operations have been correctly computed. As a consequence, the proof system generates only one proof for the whole computation, regardless of the number of sub-computations considered. Due to the efficiency requirement for VC schemes - verifying should be more efficient than computing - such embedding would a priori meet the efficiency objectives. However, a technical difficulty remains: all the VC schemes require the algorithms that will be verified to be expressed as circuits. Therefore, expressing the verification algorithm of the specialized scheme may not be as efficient as expressing the computation directly as a circuit and running it in the general-purpose VC scheme. We nonetheless propose several schemes for which the verification algorithm is more efficient than running the computation, even when expressed as a circuit. In particular, we propose an embedding of the sum-check protocol [LFKN90] that might be of independent interest. As an application of this new system, we implement a verifiable 2-layer neural network and show that our VC system improves the proving time compared to running the entire 2-layer neural network inside the GVC scheme. Thanks to this new scheme, we can run a verifiable 2-layer network with previously unreachable parameter size.

Verifiable Document Redacting [CHK17]

The first zk-SNARK application we propose addresses the following problematic: is it possible to perform modifications on a document that has been authenticated by a signature while keeping a notion of authenticity regarding this document? A primitive called *redactable signature*[JMSW02, SBZ01, BBD+10] solves this problem, however it does not scale well in terms of signature size. Indeed, the redactable signature considers a message as a set of blocks that can be redacted separately and if the message has n blocks, then the signature is of size at least $O(n)$. By contrast, the proof in our scheme has constant size. Hence, if the document we want to redact is a large image for which each pixel can potentially be redacted, the difference is significant. The protocol we propose involves three parties: a document issuer who is in charge of generating and authenticating the initial document, the client who receives the document from the document issuer and the service provider who receives a redacted document and verifies its validity. In detail, the document issuer produces the original document, computes a hash value from the document and a random value and signs the hash. Then he passes the document, its signed hash and the random value used in the hash computation process to the client. The client can then redact some parts of the original document and keeps track of each modification performed as a set of blocks that have been modified. Once this operation is done, the client computes a proof that will provide authenticity to the redacted document. Finally, the client passes the redacted document, the hash value and the set of locations of modification to the service provider. The service provider can verify the authenticity of the redacted document: he checks that the hash value is authentic thanks to its signature; if the verification passes, the hash value and the redacted document are given as inputs to the proof verification algorithm. If the signature of the hash and the proof verification both pass, the service provider can be confident that the document is authentic. The proof produced by the client is computed thanks to verifiable computation and proves that:

- there exists a document that hashes into the given hash,
- the only differences between the original document and the redacted one lie in the set of modification locations.

The proof does not reveal information about the original document thanks to the zero-knowledge property of the verifiable computation scheme. We then define security properties for our scheme:

- The scheme should be *private*: an adversary who is only in possession of the redacted message and its proof cannot recover information about the redacted parts of the message.
- The scheme should be *unforgeable*: an adversary who is not in possession of the original message cannot create a redacted document and a proof that will be accepted by the verifier.

We formalize these properties and prove that, as long as the signature scheme is secure (unforgeable under chosen message attacks) and the verifiable computation scheme is a secure zk-SNARK, our verifiable redacting document scheme is secure. We also provide experimental results of our construction, showing that the proving time a compatible with a practical use. We built a proof of concept of the verifiable redacting document scheme and presented it in the final review of the H2020 European project TREDISEC.

Privacy Preserving Biometric Commitment

The second application we propose is related to biometrics. Biometric systems enable to measure biological characteristics such as irises, fingerprints or face. Their advantage over authentication systems relying on passwords is that they suppress the need to remember any data to successfully authenticate. After a phase of collection of the biometrics by the mean of sensors, signal processing algorithms extract features from the data. A digital format compatible with an automated usage is obtained, called a biometric template. This template can be stored for a later comparison. There is variability in the biometric feature extraction process due to external conditions such as light, moisture or the sensor used for the capture. This is why feature extraction algorithms are designed such that two templates coming from the same individual are "close" while two templates coming from different individuals are "distant", these notions being related to a distance (e.g. Hamming or Euclidean, depending on the biometric modality and on the feature extraction algorithm). A threshold has to be defined to discriminate the templates belonging to the same individual or not: if the distance is lower than the threshold then the templates are supposed to belong to the same individual. If the distance is superior to the threshold, the templates come from different individuals. However, in real life, two different individuals may have a matching score lower than the threshold, which is a *false acceptance*, while two templates coming from the same individual may have a distance superior to the threshold, which is a *false rejection*. The performance of a biometric system are measured in terms of false acceptance rate (FAR) and false rejection rate (FRR). Note that FAR and FRR are linked to the threshold: if the threshold increases there are more matching pairs so the FAR increases and the FRR decreases. If the threshold decreases, there a less matching pairs so the FAR decreases and the FRR increases.

A biometric authentication proceeds in two phases: during the *enrollment*, a user presents his biometric modality in front of a sensor and a feature extraction algorithm turns the capture into a biometric template. This template is stored as a reference value along with an identifier *id* by the authentication server. During the *authentication* phase, a user who claims to be registered extracts his biometric template and sends it along with his claimed *id*. The server retrieves the reference template corresponding to *id* in its database, compares the reference template and the fresh one and decides if the user is who he claims to be. However, the fact that biometric modalities are irrevocable makes the authentication process complicated: if the biometric templates are stored without being encrypted, any breach in the database compromises the privacy and security of the individuals involved in the authentication system. On the other hand, if the template are encrypted, the inherent variation between several acquisition prevents from performing comparison with the encrypted templates.

In 1999, Juels and Wattenberg [JW99], inspired by the storage of password in UNIX systems, proposed a system to securely store biometric template. Their idea was to replace the template itself by a commitment on this template for the storage. To realize this commitment, Juels and Wattenberg relied on error correcting code theory to be able to deal with the variability of templates coming from the same individual, as long as the variation is small. They called their new primitive a *fuzzy commitment* and showed how these commitments can be leverage to perform authentication. It is important to note that, even if Juels and Wattenberg's fuzzy commitment scheme avoids the storage of biometric templates in clear, the user still has to send his template in clear when an

authentication is performed. Moreover, from a successful authentication, the server can also retrieve the reference template. Hence we propose a new fuzzy commitment protocol, that builds on the 'commit and prove' paradigm, where no template is sent in clear to the authentication server. In our scheme, the authentication server receives a classical commitment and an identifier during the enrollment phase. In the authentication phase, the user captures a fresh biometric template, computes a commitment on this template and performs a matching between his reference and fresh template. Using a zk-SNARK scheme, he then computes a proof that the templates match and that the commitments indeed open to the templates. The user sends the claimed identifier, the commitment on the fresh template and the proof. Using the identifier, the authentication server retrieves the reference commitment and calls the verification algorithm of the zk-SNARK scheme with inputs the two commitments and the proof. If the verification passes, the user is authenticated. We define security properties for our fuzzy commitment scheme:

- The scheme is correct if it accepts a proof that has been honestly computed with a template whose distance from the reference one is inferior to the threshold.
- The scheme is hiding if no useful information about the template can be extracted from the fuzzy commitment.
- The scheme is strongly binding if the fuzzy commitment cannot be opened by the sender successfully except if the template freshly captured has a distance to the reference template inferior to the threshold.

We prove that our fuzzy commitment scheme is secure as long as the underlying zk-SNARK scheme and the commitment scheme are secure. We also give experimental results (using the `libsnark` library) that show the scheme is of practical use. Moreover, from our fuzzy commitment scheme we define a privacy-preserving biometric authentication scheme. Here, the scheme is private in the sense that, when the protocol ends, the authentication server has not gained any useful information about the user's reference and fresh templates except the fact they match.

Organisation of the manuscript

The first part of this thesis presents the background required to understand verifiable computation protocols: different proof systems and their properties are described in chapter 2 and useful tool for the sequel are given. Chapter 3 describes the different VC state-of-the-art schemes within the three categories mentioned above, namely interactive proofs, interactive arguments and non-interactive arguments. A comparison is therefore made between the different state-of-the-art schemes regarding several metrics, such as the size of the proof, the proving and verifying costs and the expressiveness of the schemes.
The second part of this thesis presents in chapter 4 our first contribution: a new VC scheme tailored to efficiently deal with diverse type of operations, some of them requiring an expressive VC scheme while some others require a huge amount of computations.
The third part presents in chapter 5 two schemes that leverage zk-SNARKs schemes to provide new instantiations: a primitive that is close to redactable signatures [JMSW02] is described in Section 5.2 and a fuzzy commitment scheme [JW99] in Section 5.3.

Chapter 2

Preliminaries: Proof Systems and Useful Tools

2.1 Proofs systems

2.1.1 Classical proofs and NP languages

Proofs play a central role in science, notably in mathematics and computer science. In these disciplines, the common representation we have of a proof is a static string that contains a sequence of logically chained arguments and that can be verified by anyone. In computer science, the notion of proof is close to the **NP** complexity class: a language is in **NP** if given a statement x, it is easy to decide the membership of x in the language when given a polynomial-size solution for x that certifies this fact. Such certificates is also called a *witness* and can be viewed as a proof that x belongs to the considered language. An alternate and equivalent definition that better illustrate that idea is the following: a language is in **NP** if there exists a deterministic verifier that accepts a proof of membership for this language in polynomial time. More precisely:

Definition 1. *A language $\mathcal{L}$ is in **NP** if there exists a deterministic verifier V such that:*

- *if $x \in \mathcal{L}$, there exists w such that $V(x, w)$ accepts*
- *if $x \notin \mathcal{L}$, for every w, $V(x, w)$ rejects*
- *$V(x, w)$ runs in time polynomial in $|x|$*

Such a w is a witness or a certificate that $x \in \mathcal{L}$.

From the **NP** complexity class, we can define a proof system that involves two parties, a prover $\mathcal{P}$ and a verifier $\mathcal{V}$. The verifier and the prover have a common input, denoted by x. Without lost of generality, we assume that $\mathcal{P}$ wants to prove that the common input x belongs to a language $\mathcal{L} \subseteq \{0,1\}^*$. As a natural proof system defined from $\mathcal{L}$, $\mathcal{P}$ can simply send the witness w as a proof that the common input x belongs to $\mathcal{L}$. By the definition of $\mathcal{L}$ being in **NP**, the verifier can efficiently the claimed membership.

The two main security properties that proof systems should reach are *completeness* and *soundness*. *Completeness* basically states that if the prover behaved honestly and if the assertion is true then the verifier always accepts the assertion after the interaction.

Soundness conversely states that no cheating prover interacting with the verifier on a common input that does not belong to the language can trick the verifier into accepting the proof, except with negligible probability.

2.1.2 Interactive Proofs and Arguments

In their seminal paper [GMR85], Goldwasser, Micali and Rackoff explored a notion of proof different from the traditional one, described in the previous section. By allowing interaction between the prover and the verifier and the possibility to tie coins to establish the proof, they obtained a new proof system called *interactive proofs*. Interactive proof systems are powerful, in the sense that a large class of statements can be proved with these proof systems. Moreover, for language in **NP** the verifier of an interactive proof has less work to do than checking the witness.

An interactive proof is a game between two parties, a powerful prover and a computationally bounded verifier. The goal of the prover is to convince the verifier of the validity of some assertion. The verifier and the prover exchange messages and finally the verifier outputs a decision about the assertion. The verifier is also allowed to use randomness in its messages, which can be seen as challenges to force the prover to behave correctly. If the verifier's randomness can be shared with the prover, then the protocol is a *public coin* interactive proof. During their interactions, the verifier and the prover keep track of the exchanges in a local state and the message they send at a given round depends on the messages sent and received in all the previous rounds.

The interaction between the two parties is formalized by the notion of *strategy*. For a party, a strategy is its next move, seen as a function of all the messages received so far, its internal randomness and of the proof's common input, i.e. the assertion to be proved. A probabilistic polynomial time strategy is therefore a strategy that makes use of randomness and for which the next move can be computed in a number of steps polynomial in the size of the common input. We can now define interactive proofs:

Definition 2. *An interactive proof system for a set S is a two-party game, between a verifier executing a probabilistic polynomial-time strategy, denoted $\mathcal{V}$, and a prover that executes a computationally unbounded strategy, denoted $\mathcal{P}$ and satisfies the following properties:*

- Completeness*: for every $x \in S$, the verifier $\mathcal{V}$ always accepts after interacting with the prover $\mathcal{P}$.*
- Soundness*: for every $x \notin S$ and every strategy $\mathcal{P}^*$, the verifier $\mathcal{V}$ rejects with probability ϵ_s ($\epsilon_s < 1$) after interacting with $\mathcal{P}^*$.*

*The class of sets having interactive proof systems is denoted by **IP**.*

Several remarks can be made about definition 2. First, the probability ϵ_s, called the *soundness error*, can be made arbitrarily close to 0 by sequentially repeating calls to the proof system. Second, the number of rounds during the interaction of the prover and the verifier is polynomial in the size of the common input. Indeed, the verifier's last message is a function of all the sent and received messages and since the verifier's strategy is polynomial, this means that the number of inputs of the last message function is polynomial. Third, randomness is essential for the verifier: restricting the verifier's strategy to be deterministic leads to the complexity class **NP** ([Gol08] Ch9 - Prop

9.2). Shamir [Sha90] proved that **IP**=**PSPACE**, i.e. that the class of sets having an interactive proof coincides with the class of sets that can be decided in polynomial space. **PSPACE** class is believed to be strictly larger than the **NP** class.

Although being powerful, interactive proofs have drawbacks. For instance the number of rounds in an interactive proof is polynomial in the size of the input, which might prevents from using such proof systems if the communication bandwidth between the prover and the verifier is limited. In addition, interactive proofs protect the verifier against a computationally unbounded prover, which does not model well reality. Keeping in mind that in real life the prover would be for example a server to whom a computation would have been delegated, design a proof system secure against a computationally unbounded prover might be an overkill. Assuming that the computational power of the prover is limited enables to leverage cryptographic primitives and to achieve more goals that was possible with interactive proofs. Brassard et al. [BCC88] thus defined *interactive argument systems*, which still have to achieve correctness and soundness. The difference with interactive proof systems is that the soundness holds only against computationally bounded provers. Building on such arguments, Kilian [Kil92] built a proof system for all **NP**-statement that requires only 4 messages between the prover and the verifier, assuming that collision resistant hash functions exist.

2.1.3 Zero-knowledge Proofs

Besides defining the notion of interactive proofs, Goldwasser et al. also studied in [GMR85] the amount of information conveyed in proof systems. Indeed, in interactive proofs the prover gives information about the statement during the interactions with the verifier. Ideally, only one bit of information regarding the statement is necessary, namely a bit that conveys its correctness for the common input. Goldwasser et al. defined zero-knowledge proofs as proof systems where no information is given about the statement except its validity. To assert that the verifier could not extract information during the protocol, Goldwasser et al.'s crucial idea is that the whole conversation between the verifier and the prover could have been simulated. Therefore, they focus on the protocol view: when a verifier $\mathcal{V}$ and a prover $\mathcal{P}$ interact with common input x, $\mathcal{V}$'s output can be modeled by a random variable since each output depends on the previous messages and on $\mathcal{V}$'s internal random values, we denote by $\langle \mathcal{P}(z_P), \mathcal{V}(z_V) \rangle(x)$ this random variable. Note that $\mathcal{P}$ and $\mathcal{V}$ are allowed to have auxiliary inputs z_P and z_V.

Definition 3. *Let $\mathcal{P}$ and $\mathcal{V}$ be a prover and a verifier involved in an interactive proof with common input x. The* view *of the protocol is the distribution of the random variable defined by $\langle \mathcal{P}(z_P), \mathcal{V}(z_V) \rangle(x)$. We denote by* $\text{View}\langle \mathcal{P}(z_P), \mathcal{V}(z_V) \rangle(x)$ *such distribution.*

To formalize the fact that the verifier learns nothing about the statement to be proved from the interaction with the prover, Goldwasser et al. define an algorithm, the simulator, that on input the statement to be proved, outputs a view of a simulated protocol. A protocol is thus zero-knowledge if the output of the simulator cannot be distinguished from a view of a real execution of the protocol. Formally:

Definition 4. *Let $\mathcal{L} \subset \{0,1\}^*$ be a language. For $x \in \mathcal{L}$, let denote by R_x the set of valid witnesses for x, i.e. $R_x = \{w \text{s.t.} (x,w) \in \mathcal{L}\}$. Let R_L be the language of valid (input,witness) pairs, $R_L = \{(x,w) : x \in \mathcal{L}, w \in R_x\}$. An interactive proof system is zero-knowledge with respect to auxiliary inputs if for all probabilistic polynomial-time*

(PPT) machine V^* there exists a PPT algorithm S such that for every $x \in \mathcal{L}$, $w \in \mathcal{R}_x$, $z \in \{0,1\}^*$,

$$\text{View}\langle \mathcal{P}(w), \mathcal{V}^*(z)\rangle(x) \approx S(x,z) \tag{2.1}$$

The symbol $\approx$ means that View and the output of S are computationally indistinguishable.
The auxiliary input z represents prior knowledge: (2.1) states that even if $\mathcal{V}$ had some information before starting the protocol, he does not get more information at the end of protocol. S is called the simulator*, it runs in time polynomial in the length of x.*

Definition 4 defines *computational zero-knowledge*. Zero-knowledge proofs can also be *statistical* if the statistical distance between the two distributions is negligible. The class of languages that have zero-knowledge proofs is large: Goldreich, Micali and Widgerson [GMW86] proved, assuming that secure encryption exists, that all languages in **NP** have computational zero-knowledge proofs. Ben-Or et al. extended that result, under the same assumption, to languages in **IP** [BGG$^+$88].

2.1.4 Non-interactive arguments

Fortnow proved that only trivial languages can have an interactive perfect zero-knowledge proof [For89]. Therefore, other model removing interaction were studied: Blum et al. defined the notion of non-interactive zero-knowledge proofs [BFM88]. However the absence of communication prevents the verifier and the prover to use random challenges in the protocol. Blum et al. proposed a new model called the common reference string (CRS) model to provide access to randomness to both the verifier and the prover and proved that all languages in NP have non-interactive zero-knowledge proofs in the CRS model. Groth et al. [GOS06] proposed the first perfect non-interactive zero-knowledge argument for any language in **NP**, building their argument with a bilinear group. Their construction was later refined to reduce the size of the proof and the size of the CRS.

Definition 5. *Let $\mathcal{R}$ be an efficiently computable binary relation and $\mathcal{L}$ be the **NP**-language of statements and witnesses in $\mathcal{R}$. We denote by $\mathcal{R}_N$ the relation $\mathcal{R}$ with statements of size N. A non-interactive argument system is a triple $(G, \mathcal{P}, \mathcal{V})$, where G is the common reference string generator, $\mathcal{P}$ is the prover and $\mathcal{V}$ is the verifier.*

- *G takes a security parameter λ and some auxiliary input aux and outputs a CRS σ.*
- *$\mathcal{P}$ takes as input the CRS σ, a statement x and a witness w and outputs a proof π.*
- *$\mathcal{V}$ takes as input the CRS σ, a statement x and a proof π and outputs 1 if the argument is acceptable and 0 otherwise.*

$(G, \mathcal{P}, \mathcal{V})$ is an argument for $\mathcal{R}$ if it satisfies *completeness* and *soundness*:

Completeness: for every adversary $\mathcal{A}$ and $N = \lambda^{O(1)}$:

$$Pr[\sigma \leftarrow G(1^\lambda, N); (x,w) \leftarrow \mathcal{A}(\sigma), \pi \leftarrow \mathcal{P}(\sigma, x, w) : \mathcal{V}(\sigma, x, \pi) = 1 \text{ if } (x,w) \in \mathcal{R}_N] = 1 \tag{2.2}$$

Soundness: for all probabilistic polynomial time adversary $\mathcal{A}$ and $N = \lambda^{O(1)}$:

$$Pr[\sigma \leftarrow G(1^\lambda, N); (x,\pi) \leftarrow \mathcal{A}(\sigma) : x \notin \mathcal{L} \text{ and } \mathcal{V}(\sigma, x, \pi) = 1] \leq negl(\lambda) \tag{2.3}$$

Another construction that does not build on the CRS model is Micali's computationally sound proofs. Based on Kilian's efficient argument [Kil92], Micali defined the first non-interactive argument [Mic00] in the random oracle model. Another interesting feature of Micali's construction was the possibility to transfer the proof: once the verifier is convinced by the proof, he can pass this proof to another verifier to convince him. Micali's construction was the first argument system where size of the resulting proof was sublinear. Subsequent works tried to obtain such efficient argument without the random oracle but the only known construction that are efficient rely on non-standard cryptographic assumptions. Gentry and Wichs defined succinct non-interactive arguments (SNARG) and proved that SNARG cannot be obtained relying in falsifiable assumptions [GW11]. The definition of a SNARG add a requirement to the definition of non-interactive arguments, namely succinctness: keeping the notations of the above definition, it is required that the length of the proof π is such that:

$$|\pi| = poly(\lambda)\left(|x| + |w|\right)^{O(1)} \tag{2.4}$$

2.1.5 Proofs of knowledge

Zero-knowledge proofs enables to prove existential statements: for a statement x, there *exists* a witness w such that $(x, w) \in R$, where R is a binary relation. In proofs of knowledge, a prover claims to *know* some information. The related security properties are *knowledge completeness* and *knowledge soundness*. Knowledge completeness is very close to completeness as in definition 2, but the main difference lies in the definition of knowledge soundness, which states that if a prover can convince a verifier with non-negligible probability, then he must *know* the information on which the protocol was run. This notion is formalized by the mean of an algorithm called an *extractor* such that if the prover outputs a valid proof for a statement, the verifier can not only conclude that there exists a witness but also that the extractor can extract the witness from the prover.
Formally, a triple $(G, \mathcal{P}, \mathcal{V})$ that has the syntax of a non-interactive argument (see Section 2.1.4) is a proof of knowledge if the soundness property is replaced by the knowledge soundness property:
Knowledge soundness: For every polynomial time prover $\mathcal{P}^*$, there exists a polynomial time extractor E such that:

$$Pr\begin{bmatrix}\mathcal{V}(\sigma, x, \pi) = 1, (x, w) \notin R : \\ w \leftarrow E(aux, \sigma), (x, \pi) \leftarrow \mathcal{P}^*(aux, \sigma), \sigma \leftarrow G(1^\lambda, R)\end{bmatrix} \leq negl(\lambda) \tag{2.5}$$

A SNARG that satisfies a proof of knowledge property is therefore called a succinct non-interactive argument of knowledge (SNARK). If furthermore, this SNARK also verifies a zero-knowledge property, then it is a zero-knowledge succinct non-interactive argument of knowledge (zk-SNARK). Formal definitions and constructions of zk-SNARK schemes will be described in Section 3.3.4.

2.2 Useful Tools

2.2.1 Commitments schemes

In commitment schemes [Blu81] one player, called sender, first "commits" by submitting a value b in a concealed fashion to another player, called receiver. Later, the sender

"opens" the committed value by proving the receiver that the value stored by the latter is a hidden version of b. Commitment schemes [Blu81] fulfill two basic properties: they are *concealing* in the sense that the receiver cannot guess the value of b prior to its opening by the sender; they are *binding* in the sense that the sender cannot succeed in "opening" with a value other than b. A simple analogy is with a safe including the value b and that is locked by the sender. The receiver cannot disclose the value of b before the sender agrees to open the lock and the sender cannot substitute the value b stored in the safe with another one since the safe is kept by the receiver. Another property required is *correctness*: if the sender opens the commitment to the good message, the receiver must accept it. Depending on the scheme goal, the hiding and binding properties can be unconditional, statistical or computational but hiding and binding properties cannot be simultaneously unconditional [Fis01]. We now give syntax and security definitions of commitment schemes. We also provide some commitment examples below.

Commitment scheme syntax.

Let λ be a security parameter, a commitment scheme involves a sender A and a receiver B. The commitment algorithm is run by A and the verification algorithm is run by B.

Parameter generation: $p_k \leftarrow \texttt{KeyGen}(1^\lambda)$: on input 1^λ, the key generation algorithm `KeyGen` outputs a public key p_k.

Commit stage: $c \leftarrow \texttt{Commit}(p_k, m, r)$: the commitment algorithm `Commit` takes a message m, a public key p_k and some randomness r and outputs a commit c.

Open stage: $b \in \{0,1\} \leftarrow \texttt{Verif}(p_k, c, r, m)$: the verification algorithm `Verif` takes as input a public key p_k, a message m, a commitment c and an opening value r and outputs 1 if c opens to m and 0 otherwise.

Commitment scheme definition.

Let $(\texttt{KeyGen}, \texttt{Commit}, \texttt{Verif})$ be a commitment scheme, the three required properties for a secure commitment scheme are *correctness*, *binding* and *hiding*. We denote by *negl* a negligible function.

Correctness Let $\mathcal{M}$ denote the message space. The correctness property is formalized as follow:

$$\Pr\begin{bmatrix}\texttt{Verif}(p_k, c, r, m) = 1 : c \leftarrow \texttt{Commit}(p_k, m, r), \\ p_k \leftarrow \texttt{KeyGen}(1^\lambda), m \in \mathcal{M}\end{bmatrix} = 1$$

Binding Let $\mathcal{A}$ be a PPT adversary:

$\texttt{bind\_attack}(1^\lambda)$

$p_k \leftarrow \texttt{KeyGen}(1^\lambda)$
$(c, r_1, M_1, r_2, M_2) \leftarrow \mathcal{A}(p_k)$
if $M_1 \neq M_2$ and:
$\texttt{Verif}(p_k, c, r_1, M_1) = 1 \wedge \texttt{Verif}(p_k, c, r_2, M_2) = 1$
  return 1
else:
  return 0

The scheme is binding if for every PPT adversary:

$$\Pr\left[\texttt{bind\_attack}(1^\lambda) = 1\right] \leqslant negl(\lambda)$$

Hiding Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a PPT adversary.

$\texttt{hide\_attack}(1^\lambda)$

$p_k \leftarrow \texttt{KeyGen}(1^\lambda)$
$(aux, M_0, M_1) \leftarrow \mathcal{A}_1(1^\lambda, p_k)$
$d \xleftarrow{\$} \{0,1\}$
$r \xleftarrow{\$} \{0,1\}^{p(\lambda)}$
$c \leftarrow \texttt{Commit}(p_k, M_d, r)$
$d^* \leftarrow \mathcal{A}_2(c, aux)$
if $d = d^* \wedge M_0 \neq M_1$:
  return 1,
otherwise:
  return 0

The scheme is hiding if for every PPT adversary:

$$\Pr\left[\texttt{hide\_attack}(1^\lambda) = 1\right] \leqslant \frac{1}{2} + negl(\lambda)$$

Commitment examples.

In this section, we give some examples of commitment schemes built from different cryptographic primitives. In all the examples, A is the sender and B the receiver of the commitment.
Pedersen commitments: Let G be a cyclic group of prime order q. Pedersen commitments [Ped91] allow A to commit to a scalar $m \in \mathbb{Z}_q$. They are perfectly hiding and computationally binding under the discrete logarithm hardness assumption.

KeyGen : $p_k = (g, h)$ such that: $g \xleftarrow{\$} G$,$h \xleftarrow{\$} G$ such that g and h are generators of G.

Commit : to commit on a message $m \in \mathbb{Z}_q$, A picks $s \xleftarrow{\$} \{1, \ldots, q\}$ and computes $c = \texttt{Commit}(p_k, m, s) = g^m h^s$.

Open A sends (m, s) and B checks that: $c \stackrel{?}{=} g^m h^s$.

Commitments from hash functions: let k be a security parameter and $H \colon \{0,1\}^* \to \{0,1\}^{2k}$ be a hash function.

KeyGen : The public parameter is the description of the hash function H.

Commit : to commit on a message $m \in \{0,1\}^*$, A picks $r \xleftarrow{\$} \{0,1\}^{3k}$ and computes $c = \texttt{Commit}(p_k, m, r) = H(m \parallel r)$.

Open : A sends m and r. B checks that: $c \stackrel{?}{=} H(m \parallel r)$.

This commitment scheme can be proven secure in the random oracle model [BR05].

2.2.2 Ajtai hash function

The Ajtai hash function [Ajt96] is a hash function based on the subset sum problem. Its definition is the following:

Definition 6. *Let m, n be positive integers and q a prime number. For a randomly picked matrix $A \in \mathbb{Z}_q^{n \times m}$, the Ajtai hash $H_{n,m,q} : \{0,1\}^m \to \mathbb{Z}_q^n$ is defined as:*

$$\forall x \in \{0,1\}^m, \quad H_{n,m,q} = A \times x \mod q \tag{2.6}$$

As proved by Goldreich et al. [GGH96], the collision resistance of the hash function relies on the hardness of the Short Integer Solution (SIS) problem. Ajtai hash functions have first been used in verifiable computation by Braun et al. in [BFR+13b]. Ben-Sasson et al. [BCTV14a] then noticed that designing an arithmetic circuit that implements such hash function is easier if the parameters are chosen to fit with the underlying field of the computations.
A concrete hardness evaluation is studied by Kosba et al. in [KZM+15]. Choosing $\mathbb{F}_p$ as be the field where the computations of the arithmetic circuit take place (with p a 254-bit prime number) leads to the following parameters for approximately 100 bit of security:

$$n = 3, m = 1524, q = p \approx 2^{254}.$$

2.2.3 Bilinear pairings

A bilinear group is a tuple $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$ with the following properties:

- p is a prime number,
- $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are group of order p,
- the function $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate bilinear map, (e is called a pairing),
- g and h are respectively the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$.

If $\mathbb{G}_1 = \mathbb{G}_2$, then the bilinear group is symmetric, otherwise it is asymmetric. There exists two types of asymmetric bilinear groups, depending on the existence of an efficiently computable non-trivial homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. Asymmetric bilinear groups where there exists no homomorphism have the most efficient pairings.

All the currently known instantiations of bilinear groups that are secure for cryptography build on elliptic curves, i.e. $\mathcal{G}_1$ and $\mathcal{G}_2$ are sub-groups of an elliptic curve. If the latter elliptic curve is defined over a finite field $\mathbb{F}_r$, then the group $\mathcal{G}_T$ is a multiplicative subgroup of $\mathbb{F}_{r^k}$ for some value k linked to the elliptic curve and called the embedding degree. The pairing are implemented as the Weil pairing for the symmetric setting and the Tate pairing for the asymmetric setting [BSSC05]. We thus use additive notation for $\mathcal{G}_1$ and $\mathcal{G}_2$ and multiplicative notation for $\mathcal{G}_T$. A pairing e has therefore the following property:

$$e(P + P', Q) = e(P, Q) \cdot e(P', Q) \tag{2.7}$$
$$e(P, Q + Q') = e(P, Q) \cdot e(P, Q') \tag{2.8}$$
$$\exists P \in \mathcal{G}_1, Q \in \mathcal{G}_2 \text{ such that } e(P, Q) \neq 1 \tag{2.9}$$

From (2.7) and (2.8) it follows that: $e([a] \cdot P, [b] \cdot Q) = e(P, Q)^{ab} = e([b] \cdot P, [a] \cdot Q)$. The security of the construction basically relies on the hardness of the discrete logarithm in $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_T$.

Chapter 3

State of the Art in Verifiable Computation

Verifiable computation (VC) involves two parties: a verifier $\mathcal{V}$ and a prover $\mathcal{P}$, which are also sometimes referred as the client and the worker, recalling the client-server setting. The verifier agrees with the prover to delegate some computation, the prover carries on the calculation and sends the result to the verifier. The prover should also convince the verifier that the result is correct. Ideally, we would like to outsource a program written in a high-level language and to get insurance that the returned output of this program is correct. For that setting to be meaningful, the verification process should be efficient, more efficient than performing the computation locally, otherwise the verifier has little interest in outsourcing the computation. Moreover, the verifier should put no trust in the prover and therefore the VC scheme should provide strong guarantees against potentially malicious provers that might be cheating with the proof. There exists a large variety of dedicated VC protocols, e.g. for matrix multiplication [Fre77], polynomial evaluation [FG12, BFR13a], or database query and update [BGV11]. We focus on *general-purpose* VC schemes that typically are designed to verify the correct execution of a circuit, as discussed in Section 3.1.2. Even if they can be less efficient than the schemes dedicated to certain category of computation, the expressiveness of general-purpose schemes allows them to deal with programs involving several different categories and therefore enable outsourcing larger classes of computations.

The way to establish the correctness of the computation depends on the setting. If the prover and the verifier are allowed to communicate while establishing the correctness of the computation and no assumption is made on the prover's computational power, then verifiable computation can be obtained from interactive proofs. This connection is developed in Section 3.1. Allowing interaction but restricting the security of the system to hold against computationally bounded provers gives additional properties to the VC scheme that are unfeasible in the interactive proof setting. Such VC schemes typically are based on interactive arguments and are described in Section 3.2. Finally, there are some settings where it cannot be assumed that the verifier and the prover have the possibility to efficiently exchange messages. In such settings, the prover has to build the proof of correctness without interacting with the verifier. Such VC schemes build on the notion of non-interactive arguments and require a common reference string (CRS). Since the generation of the CRS is often costly, most non-interactive VC schemes place themselves in the pre-processing model: the CRS is generated in a first phase and its generation is not considered as part of the time that the prover takes to compute the proof. Besides, it may contain information helping the prover to build a proof more

quickly. Non-interactive arguments will be presented in Section 3.3.

Since there are plenty of approaches to build general-purpose verifiable computation schemes, there are also several metrics to measure the performance of the resulting schemes. These metrics are described in Section 3.4, strength and weaknesses are analyzed for each setting and a gap analysis is presented.

3.1 Verifiable Computation from Interactive Proofs

3.1.1 A useful interactive proof for verifiable computation: the sumcheck protocol

The Sum-Check protocol [LFKN90] enables to prove the correct evaluation of a multivariate polynomial f defined over a finite field. Suppose that f is a degree d polynomial with n variables, defined over a finite field $\mathbb{F}$ and that it has to be evaluated on the sub-cube $\{0,1\}^n$. Using the Sum-Check protocol, a prover $\mathcal{P}$ can efficiently convince a verifier $\mathcal{V}$ that he knows the value:

$$H = \sum_{t_1 \in \{0,1\}} \sum_{t_2 \in \{0,1\}} \dots \sum_{t_n \in \{0,1\}} f(t_1, \dots, t_n) \tag{3.1}$$

Note that a direct computation performed by the verifier would require at least 2^n evaluations while the Sum-Check protocol only requires $O(n)$ evaluations for the verifier. The protocol is a public coin interactive proof with n rounds of interaction during which the prover computes n univariate polynomials f_i, $i = 1, \dots, n$. These polynomials are computed from f and the degree of polynomial f_i is $\deg_i(f)$, the partial degree of f regarding the i-th variable of f. During each round, $\mathcal{V}$ generates a random field value that $\mathcal{P}$ has to integrate in the computation of the next f_i. The consistency of $\mathcal{P}$'s answer is then checked with this value. $\mathcal{V}$ is supposed to be able to compute f in one point, namely in the point $(r_1, r_2, \dots, r_n)$, defined by the random challenges r_i he has sent to $\mathcal{P}$ during the protocol rounds.
The n rounds of the protocol are summarized below:
Round 1

- $\mathcal{P}$ computes the univariate polynomial f_1 as follows:

$$f_1(x) = \sum_{t_2 \in \{0,1\}, \dots, t_n \in \{0,1\}} f(x, t_2, \dots, t_n) \tag{3.2}$$

- $\mathcal{P}$ sends H and f_1 to $\mathcal{V}$.
- $\mathcal{V}$ checks if $\mathcal{P}$ computed f_1 correctly by computing: $H = f_1(0) + f_1(1)$.
  If the equality does not hold, $\mathcal{V}$ rejects and the protocol stops.
  Otherwise, $\mathcal{V}$ chooses uniformly at random $r_1 \in \mathbb{F}$ and sends it to $\mathcal{P}$.

Round 2

- $\mathcal{P}$ computes the univariate polynomial f_2 as follows:

$$f_2(x) = \sum_{t_3 \in \{0,1\}, \dots, t_n \in \{0,1\}} f(r_1, x, t_3, \dots, t_n) \tag{3.3}$$

- $\mathcal{P}$ sends f_2 to $\mathcal{V}$.

- $\mathcal{V}$ checks if $\mathcal{P}$ computed f_2 correctly by checking: $f_1(r_1) = f_2(0) + f_2(1)$.
  If the equality does not hold, $\mathcal{V}$ rejects the response and the protocol stops.
  Otherwise, $\mathcal{V}$ picks another value $r_2 \in \mathbb{F}$ uniformly at random and sends the prover $\mathcal{P}$ the challenge.

The protocol goes on, until the last round:
Round n

- $\mathcal{P}$ computes the univariate polynomial f_n defined by:

$$f_n(x) = f(r_1, r_2, r_3, \ldots, r_{n-1}, x) \tag{3.4}$$

- $\mathcal{P}$ sends f_n to $\mathcal{V}$.
- $\mathcal{V}$ picks a random value $r_n \in \mathbb{F}$ and checks that $f_n(r_n) = f(r_1, r_2, \ldots, r_n)$.
  If the equality holds, then $\mathcal{V}$ is convinced that $H = f_1(0) + f_1(1)$ and that H has been evaluated as in (3.1).

For $H \in \mathbb{F}$, let $\mathcal{L}_H$ denote the language:

$$\mathcal{L}_H = \left\{ f \in \mathbb{F}[x_1, \ldots, x_n] \;\middle|\; \deg(f) = d, H = \sum_{t_1 \in \{0,1\}} \sum_{t_2 \in \{0,1\}} \cdots \sum_{t_n \in \{0,1\}} f(t_1, \ldots, t_n) \right\}$$

The following theorem states that the sumcheck protocol is an interactive proof and gives its soundness error:

Theorem 1. *The Sum-Check protocol is a public coin interactive proof for $\mathcal{L}_H$ with n rounds and soundness error $\epsilon_s = (n \cdot d)/|\mathbb{F}|$.*

Therefore, for a given polynomial with degree and number of variables set, the soundness ϵ_s can be made arbitrary low by choosing a large enough finite field.

3.1.2 Arithmetic circuits

The previous section described an interactive proof for the evaluation of a multivariate polynomial to a given value. Even if this language seems to include a large class of computations, it is not a priori clear how the sumcheck protocol can be linked to verifiable computing protocols that can check the correctness of any function evaluation. To establish the connection between the sumcheck protocol and verifiable computation protocols we have to deal with circuits, which are a common and convenient way to model the evaluation of a function. Moreover, all the other general-purpose VC schemes that will be described in the sequel also build on circuits. Basically, a circuit is composed of wires that carry values, it also has gates that execute a limited set of operations. The circuit can be boolean using values in $\{0, 1\}$ and the operations AND, OR and NOT or arithmetic with values in a finite field $\mathbb{F}$ and operations that are additions and multiplications over $\mathbb{F}$. The *size* of a circuit is its number of gates, the *depth* of a circuit is the longest path from the inputs to the outputs. Note that every boolean circuit can be converted into an arithmetic circuit, the size and depth of the resulting circuit increase by at most a constant factor. Circuits are very expressive: Fischer-Pippenger theorem [PF79] states that any deterministic Turing machine that runs in time $t(n)$, n being the size of the input, can be represented by a boolean circuit whose size is

no more than $t(n)\log(t(n))$. This loosely means that every computer program can *in theory* be represented as a boolean circuit, at the expense of a performance decrease. Nevertheless, representing actual programs with circuits that have size and depth small enough to implement protocols such as verifiable computation is an active research area. Several efforts have been made to represent programs as arithmetic circuits: Parno et al. [PHGR13] and Setty et al. [SVP+12] were the first to design compilers that take as input a program written in a high-level language and output arithmetic circuits. These compilers were later refined to deal with more expressive computations: Wahby et al. [WSR+15] design efficient circuits to model programs that contain data dependent control loops and random access memory while Costello et al. [CFH+15] proposed a compiler that allows state sharing between sub-computations and reduces the size of circuits containing loops and several calls to the same sub-function. Ben Sasson et al. [BCG+13] designed circuits to model a minimal random access machine, though the resulting circuits are too large to enable practical computations [WSR+15]. They also provide a low-level language to implement arithmetic circuits in the `libsnark` library [lib], which provides efficient circuits at the cost of a larger programming effort. Kosba et al. [KPS18] came up with a comprehensive comparison of programmability between existing systems. A common benchmark is the implementation of the SHA256 hash function [Nat15] as an arithmetic circuit because this function is an example of a computation that has no algebraic structure (as opposed e.g. to matrix multiplication) and thus does not easily translate into a circuit. Kosba et al. [KPS18] report 45000 gates for an implementation using Wahby et al.'s compiler, 38500 using Costello et al.'s as opposed to 26100 gates using their own compiler. Hand-optimized implementation, such as the ones using Ben Sasson et al.'s `libsnark` library report about 27000 gates [BCG+14].
The precise definition of an arithmetic circuit is the following:

Definition 7. *An arithmetic circuit over a field $\mathbb{F}$ is a directed acyclic graph in which each node has an indegree equal to 0 or 2. Node with indegree 0 represents input variables or constants. Nodes with indegree 2 either represents an addition or a multiplication in $\mathbb{F}$. Such nodes are respectively called addition and multiplication gates.*

An example of circuit is shown in Figure 3.1. The outputs of a circuit can be viewed as a function of the inputs. Denoting by $\mathcal{C}$ an arithmetic circuit and labeling $x_1, \ldots, x_n$ the inputs and $y_1, \ldots, y_m$ the outputs, one can write $(y_1, \ldots, y_m) = f(x_1, \ldots, x_n)$. An immediate recursion proves that f is a tuple of multivariate polynomials. In the sequel, for the sake of simplicity the circuit is assimilated with the function it computes. Therefore, we will often write that $\mathcal{C}$ is a function from $\mathbb{F}^n$ to $\mathbb{F}^m$ such that: $(y_1, \ldots, y_m) = \mathcal{C}(x_1, \ldots, x_n)$.

The correct execution of a circuit is linked to *circuit satisfiability*. We describe the *circuit satisfaction* problem in the case of arithmetic circuits, a similar definition exists for boolean circuits. An instance of the *arithmetic circuit satisfaction* problem is a circuit $\mathcal{C}$ that has two inputs $x \in \mathbb{F}^n$ (the *statement*) and $w \in \mathbb{F}^m$ (the *witness*) and that outputs a value $y \in \mathbb{F}^\ell$. The arithmetic circuit satisfaction asks, given a statement x, to find a witness w such that the output of $\mathcal{C}$ is 0^ℓ.

Definition 8 (Arithmetic circuit satisfaction)**.** *Let $\mathcal{C}\colon \mathbb{F}^n \times \mathbb{F}^m \to \mathbb{F}^\ell$ be an arithmetic circuit. The circuit satisfaction problem for $\mathcal{C}$ is defined by the relation $\mathcal{R}_\mathcal{C} = \{(x, w) \in \mathbb{F}^n \times \mathbb{F}^m$ such that $\mathcal{C}(x, w) = 0^\ell\}$ and its language $\mathcal{L}_\mathcal{C} = \{x \in \mathbb{F}^n : \exists w \in \mathbb{F}^m$ s.t. $\mathcal{C}(x, w) = 0^\ell\}$.*

The following theorem states that all NP problems have a reduction into an arithmetic circuit satisfaction problem:

Theorem 2 ([AB09])**.** *The arithmetic circuit satisfaction problem is NP-complete.*

Adding the possibility to provide a witness in a circuit computation leads to the class of *non-deterministic computations*. Let's look at a simple scenario that puts in evidence the power of non-deterministic computations by considering the composite number problem: given three numbers n, b_1, b_2, decide if the number n has a factor in the bound $[b_1, b_2]$. To answer such problem, one could factor n but if n and the bounds are large, since the better known factorization algorithms are sub-exponentials, the time to answer will be prohibitive. Supplying a factor of n as a witness will result in a quick answer to the initial problem: it suffices to check that the given number belongs to the bounds and that it indeed divides n, which can be done in polynomial time as opposed to the sub-exponential time required by the state of the art factorization algorithms [Sho06].

Non-determinism also allows to speed-up computations during their verification and are therefore very useful in verifiable computations as long as the VC systems accept auxiliary inputs. Suppose that you want to verify a computation that involves a division between two integer values a and b. A first attempt could be to design a circuit that implements a division algorithm between two integers, takes as inputs the two integers a and b and outputs the quotient and remainder of the division of a by b. Such circuit will have a large number of gates, at least one per bit of the divisor b. Instead, using non-determinism, one can first perform the division outside the circuit, get the quotient q and the rest r of the division and then supply these values to an arithmetic circuit that will simply check that: $a - (b \times q + r) = 0$, which only involves four operations: one multiplication, one multiplication by a constant (-1) and two additions.

From an arithmetic circuit $\mathcal{C}$ computing a function $f\colon \mathbb{F}^n \to \mathbb{F}^m$, there is a natural transformation to turn $\mathcal{C}$ into a circuit $\mathcal{C}'$ suited for the circuit satisfaction problem.

1. $\mathcal{C}'$ has $n + m$ input variables, where the m additional input variables come from the number of outputs of $\mathcal{C}$.

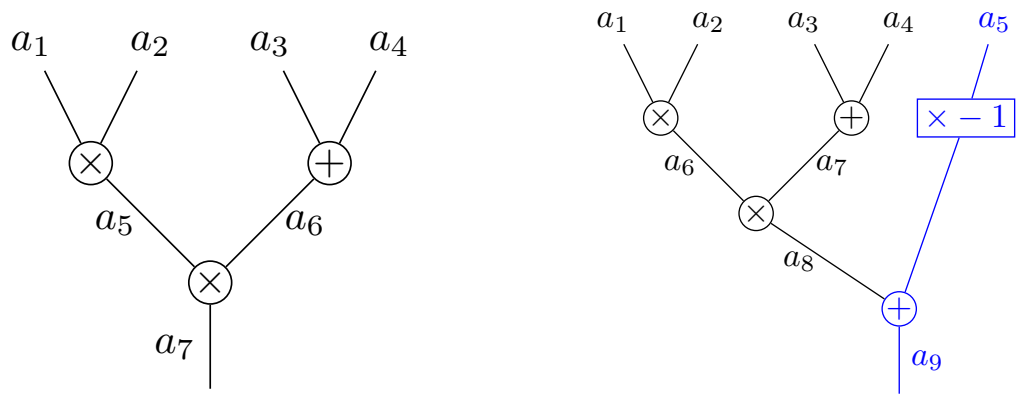
2. $\mathcal{C}'$ is such that it is a copy of $\mathcal{C}$ to which the m additional values are subtracted from the outputs of circuit $\mathcal{C}$.

Now, if a correct inputs/outputs pair of the circuit $\mathcal{C}$ is provided to $\mathcal{C}'$, it will output 0^m.
Figure 3.1 gives an example of such transformation. The original circuit, shown in Figure 3.1a, takes as inputs the values (a_1, a_2, a_3, a_4) and computes the function $f\colon (a_1, a_2, a_3, a_4) \mapsto (a_1 \cdot a_2) \cdot (a_3 + a_4)$. The output of the circuit is a_7 and a_5, a_6 are the intermediate values of the circuit.

Figure 3.1b shows the circuit *verifying* that an assignment $((a_1, a_2, a_3, a_4), a_5)$ is a correct input/output pair for the original circuit, it has been built with the transformation described above. This circuit $\mathcal{C}'$ computes $a_5 - (a_1 \cdot a_2) \cdot (a_3 + a_4)$ and is equal to 0 if a_5 is indeed the output of $\mathcal{C}(a_1, a_2, a_3, a_4)$.

3.1.3 Interactive Proofs for the Muggles (GKR)

In the proof of the result **IP**=**PSPACE** [Sha90], Shamir starts from a **PSPACE**-complete problem, namely the True Quantified Boolean Formulae (TQBF) and exhibits



(a) Original arithmetic circuit

(b) Modified arithmetic circuit suited for circuit satisfaction

Figure 3.1 – From circuit evaluation (3.1a) to circuit satisfaction (3.1b)

an interactive proof for this problem, which proves that **IP** $\supseteq$ **PSPACE**. The key idea underlying the interactive proof is to leverage *arithmetization*: the boolean formulas are represented as polynomials defined over a finite field and their correctness is checked with the sumcheck protocol (see Section 3.1.1 for details). This technique could in theory be applied to any computation by expressing the computation as an arithmetic formula and running the sumcheck protocol on it. However, even for polynomial computations, the work of the resulting prover is super-polynomial in the size of the computation input and hence the basic method is far from being practical. In 2008, Goldwasser, Kalai and Rothblum [GKR08] designed a protocol for correct circuit evaluation where the prover and the verifier are efficient. This protocol is compatible with a large class of circuits. For a circuit with the adequate structure that has S gates, Goldwasser et al. obtain an interactive proof where the prover runs in time $poly(S)$ while the verifier runs in time sub-linear in the size of the circuit: it is more efficient to delegate the computation and to verify it than to locally execute it.

At a high-level view, the protocol proceeds as follows: the verifier $\mathcal{V}$ and the prover first agree on an arithmetic circuit $\mathcal{C}$. This circuit $\mathcal{C}$ has to be layered, i.e. gates are gathered by set (layers) and gates of a particular layer are only connected to adjacent layers. The protocol proceeds layer by layer, from the output layer to the input layer. It starts with $\mathcal{P}$ sending $\mathcal{V}$ the output of layer 1, i.e. the computation output, which $\mathcal{V}$ cannot verify unless he executes the whole computation. Instead, $\mathcal{V}$ and $\mathcal{P}$ interact and reduce $\mathcal{P}$'s claim about the output to another claim about layer 2. The reduction is done in the sense that the first claim is true as long as the second one is true. The claim reduction is achieved thanks to the sumcheck protocol (see Section 3.1.1). Since $\mathcal{V}$ cannot verify the latter claim, a new interaction starts and results in a new claim about layer 3. The protocol proceeds until the interaction of $\mathcal{P}$ and $\mathcal{V}$ results in a claim about the input layer. This claim can ultimately be verified by $\mathcal{V}$: if the verification passes, $\mathcal{V}$ can accept the initial claim and the output of the computation is correct.

Goldwasser et al.'s main theorem is stated below. It applies to *logspace uniform* circuits, which are circuits that can be described by a Turing machine using logarithmic space.

Theorem 3 ([GKR08])**.** *Let $\mathcal{L}$ be a language that can be computed by a family of logspace uniform boolean circuits of size $S(n)$ and depth $d(n)$. $\mathcal{L}$ has an interactive proof with $\mathcal{P}$'s running time being $\text{poly}(S(n))$, $\mathcal{V}$'s running time being $n \cdot \text{poly}(d(n), \log S(n))$ and space $O(\log S(n))$. Moreover, the communication is $d(n) \cdot \text{polylog}(S(n))$.*

Further focusing on this protocol, we need to explain what the claims about layers are and how a layer's claim is reduced into another one's. For each layer of the circuit, Goldwasser et al. define a function that expresses the output of the layer as a function of the layer input. The resulting function is a polynomial since the gates of a layer can only be addition or multiplication gates. Cormode et al. [CMT12] denote that function a *wiring predicate*. To increase the possibility to catch a misbehaving prover, the verifier asks the prover to compute a low-degree extension from the wiring predicate. The next section defines low-degree extensions and gives useful related results for the sequel of the discussion.

Low-degree extensions

Low-degree extensions allow to apply the Sum-Check protocol to polynomials defined over some finite set included in the finite field where all the operations of the protocol are performed.
Let $\mathbb{F}$ be a finite field. For any d-variate polynomial P, we denote by $\deg_i(P)$ the degree of P in variable i.

Definition 9. *Let $\mathbb{H}$ be a subset of $\mathbb{F}$ and $f : \mathbb{H}^d \to \mathbb{F}$ be any d-variable function mapping to $\mathbb{F}$. A d-variate polynomial g over $\mathbb{F}$ is said to be an* extension *of f if:* $\forall x \in \mathbb{H}^d,\ g(x) = f(x)$.
An extension g of f is a low-degree *extension if:* $\deg_i(g) < |\mathbb{H}|, \forall i \in \{1, \ldots, d\}$. *It is* multilinear *if:* $\deg_i(g) \leqslant 1,\ \ \forall i \in \{1, \ldots, d\}$.

A multilinear extension (MLE) of a function f is therefore a polynomial that is an extension of f and has degree at most 1 in each variable. We now give a theorem that will be useful for the sequel of this section. Taking $\mathbb{H} = \{0, 1\}$, we have:

Theorem 4. *Let $f : \{0,1\}^d \to \{0,1\}$ be a function and $\mathbb{F}$ be a finite field. The function f has a unique multilinear extension over $\mathbb{F}$.*

We will hereafter denote by $\tilde{f}$ the MLE of f. Using Lagrange interpolation, an explicit expression of a MLE can be obtained as follows:

Lemma 1. *Let $f : \{0,1\}^d \to \{0,1\}$. Then $\tilde{f} : \mathbb{F}^d \to \{0,1\}$ has the following expression:*

$$\tilde{f}(x_1, \ldots, x_d) = \sum_{w \in \{0,1\}^d} f(w)\chi_w(x_1, \ldots, x_d) \tag{3.5}$$

In equation (3.5), χ_w is defined for all $w = (w_1, \ldots, w_d)$ as:

$$\chi_w(x_1, \ldots, x_d) = \prod_{i=1}^{d} \Big(x_i w_i + (1 - x_i)(1 - w_i)\Big) \tag{3.6}$$

Wiring predicate

As mentioned above, in order to build their efficient interactive proof system, Goldwasser et al. [GKR08] leverage the layered structure of the circuit and reduce the correctness of the circuit execution to a sequence of claims about the layers, the truth of each claim being related to the truth of the next one, until a last claim about the input layer. This last claim can finally be checked by the verifier since it only involves the inputs of the computation.

Let layers be labeled from 0 to d, 0 for the output layer and d for the input layer. The main component of the interactive proof proposed by Goldwasser et al. is, for each layer i, a claim made about the low-degree extension of a function that carries information regarding the inputs and outputs of layer i.

In detail, let W_i denote the function that takes as input a binary gate label and outputs the corresponding gate's value at layer i. Without loss of generality, we assume that the circuit has the same number of gates S at each layer and that gates at layer i are labeled $\{g_{i,0}, g_{i,1}, \ldots, g_{i,S-1}\}$. We denote by s the value $s = \log_2(S)$. Then: $W_i : \{0,1\}^s \to \mathbb{F}$.

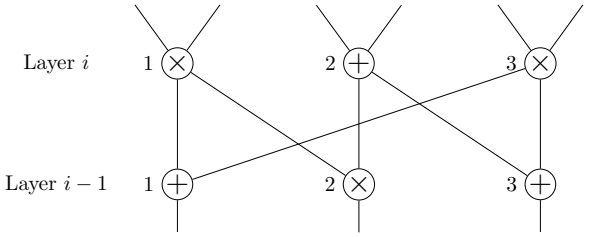


Figure 3.2 – Relation between gates of two different layers of a circuit

Figure 3.2 depicts a sample circuit with two layers, denoted i and $i-1$. Using the notations above, we have: $W_{i-1}(2) = W_i(1) \times W_i(2)$: the output value of the gate labeled 2 at layer $i-1$ is the product of the output value of the gates labeled 1 and 2 from layer i.

All the gates of a layer i are linked to two gates of the previous layer by a similar relation, involving a multiplication or an addition. Goldwasser et al. use wiring predicates to obtain an expression of W_i that carries the input/output relations of the whole layer gates. To explicit that relation, we define two functions add_i, $mult_i$ for layer i and a layer-independent function eq.

$$\begin{aligned} add_i : \{0,1\}^{3s} &\longrightarrow \{0,1\} \\ (j,k,\ell) &\longmapsto \begin{cases} 1 \text{ if } g_{i-1,j} = g_{i,k} + g_{i,\ell} \\ 0 \text{ otherwise} \end{cases} \end{aligned} \tag{3.7}$$

$$\begin{aligned} mult_i : \{0,1\}^{3s} &\longrightarrow \{0,1\} \\ (j,k,\ell) &\longmapsto \begin{cases} 1 \text{ if } g_{i-1,j} = g_{i,k} \times g_{i,\ell} \\ 0 \text{ otherwise} \end{cases} \end{aligned} \tag{3.8}$$

$$\begin{aligned} eq : \{0,1\}^{2s} &\longrightarrow \{0,1\} \\ (x,y) &\longmapsto \begin{cases} 1 \text{ if } x = y \\ 0 \text{ otherwise} \end{cases} \end{aligned} \tag{3.9}$$

The value $add_i(j,k,\ell)$ equals one if the gate labeled by j at layer $i-1$ is the sum of the gates labeled k and ℓ of layer i while $mult_i(j,k,\ell)$ encodes the same information for gate multiplication. The functions add_i, mul_i and eq allow to get an explicit and global expression that links W_i and W_{i-1}. For $z \in \{0,1\}^s$, we have:

$$W_{i-1}(z) = \sum_{(j,k,\ell)\in\{0,1\}^{3s}} eq(z,j) \cdot \Big(add_i(j,k,\ell) \cdot (W_i(k) + W_i(\ell)) + mult_i(j,k,\ell) \cdot W_i(k) \cdot W_i(\ell) \Big) \tag{3.10}$$

Equation (3.10) states that at layer $i-1$, the value of the gate labeled z is the sum or the product of gates from layer i whose output is the input of the gate labeled z. The function eq acts here like a selector and allows to have a single expression for the whole layer even if add_i and $mult_i$ are functions acting at the gate level. The low-degree extension of (3.10) finally enables the reduction from a layer to the previous one. The verifier sends the prover a random value of $\mathbb{F}$ and then they engage in a sumcheck protocol about the evaluation over $\{0,1\}^{3s}$ of the polynomial $f_i(z) := \widetilde{eq}(z,j) \cdot \Big(\widetilde{add}_i(j,k,\ell) \cdot (\widetilde{W}_i(k) + \widetilde{W}_i(\ell)) + \widetilde{mult}_i(j,k,\ell) \cdot \widetilde{W}_i(k) \cdot \widetilde{W}_i(\ell) \Big)$. Cormode et al. [CMT12] estimate that a naive implementation of the prover would give a complexity of $\Omega(S^3)$, i.e. the work of the prover is at least cubic in the size of the circuit. Goldwasser et al. also note that the functions $\widetilde{add}_i$ and $\widetilde{mult}_i$ are *circuit-dependent* but not *input dependent*. Therefore, some work can be saved by performing offline computations to pre-compute the latter functions. Indeed, the verifier and the prover are supposed to agree on a given circuit on which they will run the protocol. Both can then compute the functions $\widetilde{add}_i$ and $\widetilde{mult}_i$ before the circuit is evaluated and save time during the interactions.

In the sequel, we will denote Goldwasser et al.'s protocol by GKR. Next section describes the efforts made to actually implement the GKR protocol, mainly through some efficiency improvements. Furthermore, some optimizations allowed to decrease the prover's work, resulting in an optimal prover for certain class of circuits.

3.1.4 Implementation of the GKR protocol and later optimizations

In 2012, Cormode Mitzenmacher and Thaler published a paper [CMT12] where they studied Goldwasser et al.'s protocol with a practical purpose in mind. The paper contains refinement that were required to be able to implement the protocol. One of the major contributions is an improvement on the prover's work. Cormode et al. notice that, for each round of the sumcheck protocol applied to the polynomial f_i (as defined in section 3.1.3), the gates at layer i and $i-1$ only contribute for a single term of the sum computed in the round. For a circuit of size S, this observation enables Cormode et al. to decrease the prover's work from $\Omega(S^3)$ to $O(S \log S)$. They also provide an implementation of their refined protocol and give several examples that are benchmarked. In a subsequent note [1], Thaler notes that the term $\widetilde{eq}$ is not necessary in (3.10), provided that the low-

[1] `http://people.cs.georgetown.edu/jthaler/GKRNote.pdf`

degree extensions are *multilinear* extensions. This observation reduces the umber of rounds needed in each application of the sumcheck protocol and the hidden constants in the $O(S \log S)$.

In a paper published at Crypto '13, by carefully studying the expression (3.10), Thaler [Tha13] notices that the polynomial f_i used by Cormode et al. is very sparse. This, on the one hand enables to improve the prover's efficiency but on the other hand prevents from reusing work over iterations of the sumcheck protocol. Choosing a new polynomial defined over a smaller domain to express the relation between W_i and W_{i-1}, Thaler succeeds in decreasing the prover's work to $O(S)$, which is optimal: the prover's work to evaluate the circuit and compute the proof increases only by a constant. However, Thaler's optimization applies to layered circuits that possesses supplementary structure. This still encompasses a large class of circuits computing non-trivial problems, notably matrix multiplication, pattern matching, frequency vector computations in data streaming or fast Fourier transform [Tha13]. The experimental results show a prover that is at least 200 times faster and a communication two times less than Cormode et al.'s implementation.

Building on Thaler's work, more optimizations were proposed for structured circuits: Wahby et al. [WJB+17b] show that if the considered arithmetic circuit is structured with multiple copies of the same sub-circuit, which models well parallelizable computations, then the prover's work can further be decreased. They show that when a sub-computation is repeated about 30 times, the work of the prover is linear in the size of the computation, with a hidden constant being approximately 10: to prove correctness of a computation, the prover only has to perform 10 times more work than if he ran the unproven computation.

3.2 Verifiable Computation from Interactive Arguments

Verifiable computation has strong links with *non-deterministic* computations because *verifying* and *performing* a computation often do not have the same complexity. A typical example is the factorization of a number: while it is difficult to *factor* n, allowing non-determinism makes the problem easy since it suffices to provide factors p and q as witnesses and to check that the given integer n is indeed the product of p and q. The factorization problem has sub-exponential complexity for the better algorithms while the verification that factors provided for a number are correct has polynomial complexity. A large class of problems can be solved when the possibility to provide a witness in a computation is added. Such problems can be modeled by the computation of *non-deterministic circuits*: a non-deterministic circuit $\mathcal{C}$ takes two inputs x and y and accepts x if there exists an input y such that $\mathcal{C}(x, y) = 1$. The input value y is the *witness* of the computation. It is sometimes referred as the *auxiliary input*, the *advice* or the *guess*. A practical advantage of non-deterministic computations is that the prover can choose any efficient algorithm to solve the problem and only has to follow a given algorithm to prove that he owns the witness of a correct result. This principle applies to interactive proofs: for instance, the multiplication of two (n, n) matrices can easily be translated into an arithmetic circuit of size $O(n^3)$ that lends itself well for Goldwasser et al.'s protocol [GKR08]. But, even if Thaler's optimal protocol [Tha13] is used, the prover still has a $O(n^3)$ work to compute the proof whereas he could prove the matrix multiplication correctness using Freivalds' algorithm [Fre77] and perform the verification in time $O(n^2)$. In the case of interactive proof protocols deriving from the GKR protocol, a problem

remains because, in order to prove the correctness of a computation, the prover has to send the witness and to run the protocol involving it. This can be problematic for several reasons. First, the witness may be too large and thus cannot be sent or its size may dramatically increase the communication cost of the interactive proof. Zhang et al. [ZGK+17] consider the outsourcing of a database into the cloud with verifiable SQL queries and show that for some queries, the witness of the computation can be as large as the whole database. Second, the prover may want to keep the witness secret as in the case of factorization: proving that one knows the factorization of an RSA modulus n has no interest if the prime factors p and q are disclosed because the security of the RSA scheme relies on the difficulty to factor the modulus.

Such limitations can be overcome by weakening the computational power of the prover in the soundness requirement of Definition 2. This relaxation on security properties enables to leverage cryptographic techniques and therefore to get additional properties for the considered schemes, such as efficient non-deterministic computation or public verifiability of the proof. Brassard et al. [BCC88] define *argument* systems by restricting the soundness to hold against computationally bounded provers. Note that a prover with super-polynomial computing power can then make the verifier accept the proof of a false statement. On the next section, we describe the evolution of arguments from their definition to their implementation as verifiable computing systems.

3.2.1 Interactive Arguments

Definition 10. *An interactive argument system for a set S is a two-party game, between a verifier, denoted $\mathcal{V}$, and a prover denoted $\mathcal{P}$, satisfying the following properties:*

- ***Completeness**: The prover $\mathcal{P}$ runs in polynomial time in the size of the common input. For every $x \in S$, there exists an auxiliary input w_x so that the verifier $\mathcal{V}$ always accepts after interacting with the prover $\mathcal{P}(w_x)$ on common input x.*
- ***Soundness**: For every probabilistic polynomial time prover $\mathcal{P}$, for all $x \notin S$ and for all $w \in \{0,1\}^*$, the verifier $\mathcal{V}$ rejects with probability ϵ_s ($\epsilon_s < 1$) after interacting with $\mathcal{P}$ on common input x.*

The first asymptotically efficient arguments build on the probabilistically checkable proofs (PCP) theorem [AS98], which notably states that for every NP language, there is an encoding of the proof that can be probabilistically verified by querying only a small number of bits. To satisfy this requirement, the proof encoding must add redundancy. More precisely, the length of the encoded proof in the PCP theorem is polynomial in the size of the original proof and the number of queries is constant (more queries decrease the soundness error of the proof system). One could envision building an interactive proof system using PCPs but in the PCP theorem [AS98] the soundness holds only if the proof is immutable: a cheating prover could adapt his answer to the verifier's queries to make them consistent with the protocol. Therefore, in a proof system based on PCP the prover should send the whole PCP to the verifier. However, the size of the PCP is superior to the size of the computation witness and thus the resulting proof system would be inefficient.

Kilian [Kil92] and Micali [Mic00] overcome that problem by defining efficient arguments from PCPs. Instead of sending the PCP, the prover rather commits to the proof

obtained with the PCP theorem and sends this commitment to the verifier. Then, the verifier makes queries for a few bits of the proof and the prover decommits those bits. Leveraging the commitment to the previously received PCP, the verifier can then check the prover's answer without the need for the prover to decommit the whole proof. The latter schemes are only relevant if the commit to the PCP has a small size, which is reached by the mean of Merkle trees [Mer87].

3.2.2 Ishai et al. efficient arguments and later optimizations

Albeit asymptotically efficient, Kilian's or Micali's constructions have no practical interest due to the time required to compute a PCP or the space required to store it. Ishai et al. [IKO07] proposed another argument system with worse asymptotic complexity but also potentially better efficiency for practical instances. Their idea is to give up on having a polynomial-size proof: the proof is described as a function that can have exponential size and a proof query is simply the evaluation of such function. In details, let $\mathcal{P}$ be a prover and $\mathcal{V}$ be a verifier. Let us consider a circuit $\mathcal{C}$ for which $\mathcal{P}$ wants to prove he knows a correct assignment (x, w) and denote by W a transcript of the evaluation of $\mathcal{C}$: W is a string supposed to contain constraints that check if x has really been applied as input, if the input/output relations of each gate of $\mathcal{C}$ are correct and if the output of $\mathcal{C}$ is indeed 1. The PCP function f_W will then be the Walsh-Hadamard encoding of W [AB09] and the PCP itself – that we denote by π – consists of all the evaluations of f_W. Note that f_W is a linear function and that, if $\mathcal{C}$ has s gates, $f_W : \mathbb{F}^{s^2+s} \to \mathbb{F}$ and π has size $|\mathbb{F}|^{s^2+s}$. In Ishai et al.'s scheme, that we will denote by IKO, the verifier then makes queries whose goal is to check that:

i) f_W is indeed a linear function,

ii) π contains the evaluation of a Walsh-Hadamard encoding of W,

iii) W satisfies all the constraints of circuit $\mathcal{C}$

The checks are performed thanks to an additively homomorphic encryption scheme E: the verifier $\mathcal{V}$ picks a random vector r and sends $\mathcal{P}$ the encryption $E(r)$. The prover can then evaluate f_W at r and get $E(f_W(r))$. Since $\mathcal{P}$ has no information about r, the computation of $E(f_W(r))$ serves for the purpose of a commitment on function f_W. The subsequent queries also leverage the homomorphism to check that the above mentioned properties are satisfied.

IKO as a VC system

Based on the IKO protocol, Setty et al. [SMBW12] achieve the first implementation of a VC system. They instantiate the homomorphic encryption scheme with El-Gamal encryption [Gam84] and refine IKO protocol to deal with arithmetic circuits instead of boolean circuits in the original protocol. They also perform verification on batched input to amortize the cost of the PCP queries. Finally, they provide a compiler that takes the description of a circuit in a high-level language and output the set of constraints that will define the transcript W of a correct computation. Subsequent works [SVP+12, BFR+13b, WSR+15] refine the resulting VC system by adding expressiveness through the optimization of the compiler or the one at the linear function whose evaluation defines the PCP π. This latter enhancement builds a linear PCP from Gennaro et al.'s

Quadratic Arithmetic Programs (QAP) [GGPR13] (see Section 3.3.2 for details on QAP construction).

3.2.3 Interactive Arguments from CMT

Zhang et al.'s argument [ZGK$^+$17]

Recently, Zhang et al. [ZGK$^+$17] proposed an extension to the CMT protocol (see Section 3.1.4), turning it into an interactive argument. When studying the outsourcing of a database compatible with verifiable queries, Zhang et al. raised a need that was the same as the one described in Section 3.2.1, namely to efficiently prove the correctness of a non-deterministic computation without sending the witness. Zhang et al. first note that in the interactions of the CMT protocol, the verifier does not need to know the witness except in the last round of interactions where he only needs to perform computation with the *evaluation* of the multilinear extension of the witness at a random value. Zhang et al. therefore define a polynomial commitment scheme and force the prover to commit on the multilinear extension $\tilde{w}$ of the witness w, which is indeed a polynomial. Later, during the CMT protocol, upon receiving the verifier's challenge r, the prover will send him the value $\tilde{w}(r)$. $\mathcal{V}$ can then check that this value is indeed the evaluation of the committed polynomial $\tilde{w}$ at r. The commitment enables the verifier to work on $\tilde{w}(r)$ without ever seeing the witness w. It also gives the verifier confidence that the prover cannot cheat about the witness, as long as the cryptographic assumption on which the commitment scheme relies holds. Zhang et al.'s polynomial commitment scheme leverages bilinear pairing for the opening and relies on two cryptographic assumptions: a generalization of Groth's knowledge of exponent assumption [Gro10] and a strong Diffie-Hellman assumption [BB04]. Note also that the scheme requires a pre-processing phase where the parameters of the polynomial commitment scheme are computed. This phase consists of a *trusted setup*, in the sense that the parameter generation requires some randomness that must be discarded once the setup phase ends. An adversary in possession of such randomness is able to break the soundness of the scheme and thus to forge a proof for bogus results that will nonetheless be accepted. Assume that the witness of the scheme, here the database to be outsourced, is denoted by w. Then Zhang et al. show that:

- The setup phase runs in time linear in the size of the witness.
- The prover runs in time quasi-linear in the size of the circuit and of the witness.
- The verifier runs in time sub-linear in the size of the circuit and of the witness.

Wahby et al.'s argument [WTS$^+$18]

As Zhang et al. [ZGK$^+$17], Wahby et al. [WTS$^+$18] leverage commitment schemes to turn CMT into an argument. Although being related, Wahby et al.'s approach brings several improvements: their scheme is zero-knowledge, relies on weaker and standard cryptographic assumptions and results in a more efficient argument scheme. They start from an slightly improved version of CMT [WJB$^+$17b] and use several commitment schemes that have a homomorphic property and for which the prover can prove the opening in zero-knowledge. These commitment schemes allow to implement a zero-knowledge version of the sumcheck protocol (see Section 3.1.1). Since this protocol

involves successive claim reductions about the arithmetic circuit from the output layer to the input layer, Wahby et al.'s modification of CMT is therefore zero-knowledge. In detail, Wahby et al. use Pedersen's commitment scheme [Ped91] and prove in zero-knowledge the knowledge of the commitment opening, that two commits open to the same value and that a committed value is the product of two committed values. During the sumcheck protocol, the prover sends commitments to the coefficients of the intermediate polynomials, denoted f_i. In Wahby et al.'s protocol, the intermediate polynomials involved in the sumcheck protocol have degree 3: the prover thus commits to 4 values. From those commitments, the verifier $\mathcal{V}$ has to check that: $f_i(0) + f_i(1) = f_{i-1}(r_{i-1})$. Since $\mathcal{V}$ knows r_{i-1}, this comes down to computing the left hand and right hand sides of the previous equality and to engage with the prover in a zero-knowledge proof if the two obtained commitments open to the same value. The first round of interaction in the sumcheck is a little different but the techniques rely on the same principles. Wahby et al.'s argument system is particularly efficient if the circuit considered for verification has some regularity, e.g. the circuit to verify has N identical sub-computations. Besides, it relies on classic cryptographic assumptions (i.e. the Decisional Diffie Hellman assumption [CS98]). For circuits that reproduce the same sub-computation N times, $N \geq 30$:

- the communication costs are sub-linear in the size of the circuits and of the witness,
- the prover runs in time linear in the size of the circuit (with small constants),
- the verifier runs in time sub-linear in the size of the circuits and the witness.

3.3 Verifiable Computation from Non-interactive Arguments

There are settings where it is difficult to assume that a verifier $\mathcal{V}$ and a prover $\mathcal{P}$ have access to a channel over which they can efficiently communicate. This raises the need for defining non-interactive verifiable computation where the only interactions during the protocol are $\mathcal{V}$ sending the desired input for the computation and $\mathcal{P}$ sending back the result of the computation and the proof of correctness.

A first possibility is to use the Fiat-Shamir transformation [FS86] to turn a public-coin interactive argument into a non-interactive argument. The idea of this transformation is to ask the prover to compute himself the random values sent by the verifier during the protocol: the prover replaces the uniformly random challenges sent by the verifier with challenges he computes applying a public hash function to the transcript of the protocol so far. The prover then sends the whole protocol transcript, which can be verified by recomputing the challenges with the same hash function. This method has been proved secure in the random oracle model [PS00].

However, this methodology has a drawback: the resulting proof is at least equal to the amount of communication produced during the interactive argument. Therefore, other approaches try to directly define non-interactive arguments tailored for verifiable computation. Gennaro et al. were the first to formally define Verifiable Computation (VC) and to propose a non-interactive scheme that satisfies the VC requirements. Their construction leverages fully homomorphic encryption and therefore does not fit a practical and efficient verifiable computing scheme. The definition was successively refined [PRV12, GGPR13, PHGR13], for the sequel we rely on Parno et al.'s definition [PHGR13].

3.3.1 Definition

Definition 11 (VC scheme)**.** *Let F be a function, expressed as an arithmetic circuit over a finite field $\mathbb{F}$ and λ be a security parameter.*

- $(EK_F, VK_F) \leftarrow \texttt{KeyGen}(1^\lambda, F)$*: the randomized algorithm* `KeyGen` *takes as input a security parameter and an arithmetic circuit and produces two public keys, an evaluation key EK_F and a verification key VK_F.*
- $(y, \pi) \leftarrow \texttt{Prove}(EK_F, x)$*: the deterministic* `Prove` *algorithm, takes as inputs x and the evaluation key EK_F and computes $y = F(x)$ and a proof π that y has been correctly computed.*
- $\{0,1\} \leftarrow \texttt{Verify}(VK_F, x, y, \pi)$*: the deterministic algorithm* `Verify` *takes the input/output (x, y) of the computation F, the proof π and the verification key VK_F and outputs 1 if $y = F(x)$ and 0 otherwise.*

The desired security properties for a publicly verifiable VC scheme are the following:

- *Correctness*: for any function F and any input x:

$$Pr\left[\begin{array}{c}(EK_F, VK_F) \leftarrow \texttt{KeyGen}(F, 1^\lambda);\\ (y, \pi) \leftarrow \texttt{Compute}(EK_F, x);\\ \texttt{Verify}(VK_F, x, y, \pi) = 1\end{array}\right] = 1$$

- *Soundness*: for any function F and any probabilistic polynomial-time adversary $\mathcal{A}$,

$$Pr\left[\begin{array}{c}(u, y, \pi) \leftarrow \mathcal{A}(EK_F, VK_F) :\\ F(u) \neq y \text{ and}\\ \texttt{Verify}(VK_F, u, y, \pi) = 1\end{array}\right] \leqslant \texttt{negl}(\lambda)$$

- *Efficiency*: the cost of `Verify` is lower than the cost of executing F.

Parno et al. [PHGR13] consider that since `KeyGen` is only performed once for the lifetime of the VC system, its costs is amortized over all future instances of the problem and has not to be taken into account for the efficiency of the system. In constrast, some later works, e.g.[WJB+17b], consider that the `KeyGen` phase has to be included in the efficiency metric.

3.3.2 Main tool: Quadratic Arithmetic Programs (GGPR13)

Assume that you want to check the correctness of a computation, expressed as an arithmetic circuit. Given an assignment of the input variables and the output value, a straightforward solution is to evaluate the circuit and to verify that the circuit output is indeed the given output value. But is it possible to check correctness while doing less computations ? Quadratic arithmetic programs (QAP) enable to perform the latter task more efficiently than redoing the whole computation, they reduce the satisfiability of an arithmetic circuit (see Section 3.1.2) to a divisibility check between polynomials. Let us explain in detail how the reduction from an arithmetic circuit is performed. We start by giving an arithmetic circuit example and the construction of its related QAP

before formally defining QAPs and stating their properties. Consider the function f defined over $\mathbb{F}_{37}$ by:

$$f\colon (a_1, a_2, a_3, a_4) \mapsto (2a_2 \cdot (a_1 + 3) + (a_2 + a_3) \cdot a_4) \cdot a_4. \tag{3.11}$$

This function can be represented as the arithmetic circuit displayed in Figure 3.3. All the wires of the circuit are labeled: in figure 3.3 the input values are a_1, a_2, a_3, a_4, the output is a_7, while a_5 and a_6 are intermediate values.

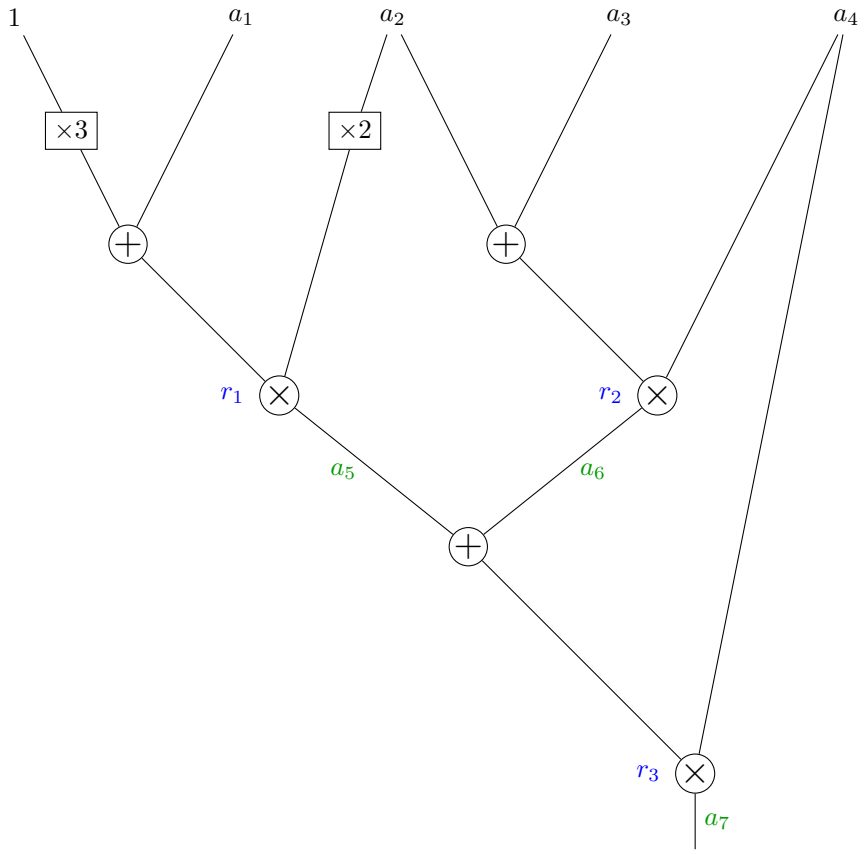


Figure 3.3 – Arithmetic circuit for $f\colon (a_1, a_2, a_3, a_4) \mapsto (2a_2 \cdot (a_1 + 3) + (a_2 + a_3) \cdot a_4) \cdot a_4$

The relation between the inputs and the outputs of the function is a multivariate polynomial whose degree and number of variables do not easily lend themselves to cryptographic operations. For instance, Ben Sasson et al. [BCG+14] report an optimized arithmetic circuit with about 30000 multiplicative gates that represents the compression function of SHA256 [Nat15]. This means that the correctness check would involve a degree 30000 polynomial for this single hash computation. Instead of dealing with the constraints between the inputs and the output of the whole circuit, Gennaro et al. [GGPR13] flatten the circuit by considering the input/output relationships of each gate, which leads to a set of quadratic equations. In the case of function f, the set is:

$$f(a_1, a_2, a_3, a_4) = (2a_2 \cdot (a_1 + 3) + (a_2 + a_3) \cdot a_4) \cdot a_4 \iff \begin{cases} a_5 = 2a_2 \cdot (a_1 + 3) & (3.12a) \\ a_6 = (a_2 + a_3) \cdot a_4 & (3.12b) \\ a_7 = (a_5 + a_6) \cdot a_4 & (3.12c) \end{cases}$$

The quadratic arithmetic programs leverage the equations (3.12) and build two polynomials from this set, one that will contain all the relationship and one that will enable to efficiently check the correctness of the relationships using polynomial division. To build these polynomials, each multiplicative gate g of the arithmetic circuit is associated to an arbitrary value r_g. Then, three polynomial families $\mathcal{V} = \{v_k(x)\}$, $\mathcal{W} = \{w_k(x)\}$ and $\mathcal{Y} = \{y_k(x)\}$ are defined from each value of the circuit – input, output or intermediate. Denoting by G the set of multiplication gates of the circuit, the polynomials are defined by the value they take on $(r_g)_{g \in G}$. Polynomials in $\mathcal{V}$ represent left inputs of gates, polynomials in $\mathcal{W}$ represent right inputs of gates while polynomials in $\mathcal{Y}$ represent outputs of gates. Table 3.1 gives all these polynomials for the arithmetic circuit described in Figure 3.3. For instance, polynomial $v_1(r_1) = 1$ means that the wire labeled by a_1 is a right input of the gate 1, associated with the value r_1. Note that even if $\mathcal{V}$, $\mathcal{W}$ and $\mathcal{Y}$ are defined only at multiplication gates, multiplications by constants and additions are taken into account: for example, $w_2(r_1) = 2$ means that a right input of the gate 1, associated with value r_1 is $2 \times a_2$. The case of addition will be considered few lines below. To see how these polynomial families enable to check if a circuit assignment is correct, let consider an assignment $(a_1, a_2, a_3, a_4, a_5, a_6, a_7)$. From the families $\mathcal{V} = \{v_i\}_i$, $\mathcal{W} = \{w_i\}_i$, $\mathcal{Y} = \{y_i\}_i$, we compute three polynomials v, w, y such that:

$$v(x) = \sum_{i=1}^{7} a_i v_i(x) \tag{3.13a}$$

$$w(x) = \sum_{i=1}^{7} a_i w_i(x) \tag{3.13b}$$

$$y(x) = \sum_{i=1}^{7} a_i y_i(x) \tag{3.13c}$$

	r_1	r_2	r_3
v_0	3	0	0
v_1	1	0	0
v_2	0	1	0
v_3	0	1	0
v_4	0	0	0
v_5	0	0	1
v_6	0	0	1
v_7	0	0	0

	r_1	r_2	r_3
w_0	0	0	0
w_1	0	0	0
w_2	2	0	0
w_3	0	0	0
w_4	0	1	1
w_5	0	0	0
w_6	0	0	0
w_7	0	0	0

	r_1	r_2	r_3
y_0	0	0	0
y_1	0	0	0
y_2	0	0	0
y_3	0	0	0
y_4	0	0	0
y_5	1	0	0
y_6	0	1	0
y_7	0	0	1

Table 3.1 – QAP polynomial family for the arithmetic circuit defined in figure 3.3

Now, using equation (3.13a) and Table 3.1, we see that: $v(r_2) = \sum a_i v_i(r_2) = a_2 + a_3$ which is exactly the fact that a_2 and a_3 are left inputs of the gate 2 associated to r_2. This also shows that additions are taken into account by the polynomial families. Next, let us define a polynomial p by $p(x) = v(x) \cdot w(x) - y(x)$. The intuition for p is that it will somewhat compress all the input-output relations at the circuit multiplication gates. Indeed, by inspection we see that evaluating p at a value associated to a multiplicative gate gives the input/output relation at this gate. In our running example, the evaluation

of p at r_2 gives:

$$p(r_2) = (\sum_{i=1}^{7} a_i v_i(r_2)) \cdot (\sum_{i=1}^{7} a_i w_i(r_2)) - (\sum_{i=1}^{7} a_i y_i(r_2)) = (a_2 + a_3) \cdot (a_4) - a_7 \qquad (3.14)$$

Therefore, if p vanishes at r_2 then: $a_7 = (a_2 + a_3) \cdot (a_4)$, which exactly describes a correct input-output relationship at gate 2. Conversely, if equation (3.12c) holds then, by relation (3.14) p will vanish at r_2.
The interest of the polynomial p is that a divisibility check is enough to check all the input-output relations mentioned above: denoting by $\{r_g \mid g \in G\}$ the values associated with each gate, we first define a polynomial t as $t(x) = \prod_{g \in G}(x - r_g)$. This polynomial will enable to verify that the circuit assignment is correct by checking that it divides p. Indeed, if the input-output relationships hold for each multiplication gate of the circuit, the polynomial p built from the $\mathcal{V}$, $\mathcal{W}$ and $\mathcal{Y}$ families will vanish at every value of the set $\{r_g \mid g \in G\}$. The fact that p vanishes at every value r_g means that $p(x)$ is divisible by the polynomial $\prod_{g \in G}(x - r_g)$, which is by construction the polynomial $t(x)$. To sum up: *The correctness of the circuit assignment comes down to check that $p(x)$ is divisible by $t(x)$.*

The polynomial p is computed by interpolating the $\mathcal{V}$, $\mathcal{W}$ and $\mathcal{Y}$ polynomials from their values at $\{r_g \mid g \in G\}$ and computing:

$$p(x) = (\sum_{i=1}^{|G|} a_i v_i(x)) \cdot (\sum_{i=1}^{|G|} a_i w_i(x)) - (\sum_{i=1}^{|G|} a_i y_i(x)) \qquad (3.15)$$

Note that, by construction, many of the polynomials in $\mathcal{V}$, $\mathcal{W}$ and $\mathcal{Y}$ are equal to zero, which is leverage to compute efficiently p from the evaluated polynomials. In our running example, setting $r_1 = 1$, $r_2 = 10$, $r_3 = 26$ and using Table 3.1, we obtain:

$v_0(x)$	$x^2 + x + 1$	$w_0(x)$	0	$y_0(x)$	0
$v_1(x)$	$25x^2 + 25x + 25$	$w_1(x)$	0	$y_1(x)$	0
$v_2(x)$	$28x^2 + 21x + 25$	$w_2(x)$	$27x^2 + 27x + 27$	$y_2(x)$	0
$v_3(x)$	$28x^2 + 21x + 25$	$w_3(x)$	0	$y_3(x)$	0
$v_4(x)$	0	$w_4(x)$	$12x^2 + 12x + 13$	$y_4(x)$	0
$v_5(x)$	$21x^2 + 28x + 25$	$w_5(x)$	0	$y_5(x)$	$25x^2 + 25x + 25$
$v_6(x)$	$21x^2 + 28x + 25$	$w_6(x)$	0	$y_6(x)$	$28x^2 + 21x + 25$
$v_7(x)$	0	$w_7(x)$	0	$y_7(x)$	$21x^2 + 28x + 25$

Quadratic arithmetic programs have been defined by Gennaro et al. [GGPR13]. As in [Gro16], we slightly modify the original notations in the characterization to ease the exposition:

Definition 12. *A quadratic arithmetic program (QAP) $\mathcal{Q}$ over a field $\mathbb{F}$ is a tuple of polynomials from $\mathbb{F}[x]$ such that: $\mathcal{Q} = \{\{v_k(x)\}, \{w_k(x)\}, \{y_k(x)\}, k \in \{0, \ldots, m\}, t(x)\}$. The size of $\mathcal{Q}$ is m and its degree is* $\deg(t(x))$.

Definition 13. *Let $f \colon \mathbb{F}^n \to \mathbb{F}^{n'}$ be a function with input variables labeled by $\{1, \ldots, n\}$ and output variables labeled by $\{n+1, \ldots, n+n'\}$. A QAP $\mathcal{Q}$ computes f if, denoting $\mathcal{Q} = \{\{v_k(x)\}, \{w_k(x)\}, \{y_k(x)\}, k \in \{0, \ldots, m\}, t(x)\}$, the following assertion is true:*

$$(a_1, \ldots, a_n, a_{n+n'}) \textit{ is a valid input/output assignment for } f$$
$$\Leftrightarrow$$
$$\exists (a_{n+n'+1} \ldots, a_m) \in \mathbb{F}^{m-n-n'} \textit{ s.t. } t(x) \textit{ divides } p(x) := (\sum_{i=1}^{m} a_i v_i(x)) \cdot (\sum_{i=1}^{m} a_i w_i(x)) - (\sum_{i=1}^{m} a_i y_i(x))$$

Theorem 5 states that for every arithmetic circuit there exists a QAP that can be efficiently built.

Theorem 5. *Let $\mathcal{C}$ be an arithmetic circuit with input from $\mathbb{F}^n$, that has s multiplicative gates, each with fan-in two, and whose output gates are all multiplication gates. There is a QAP $\mathcal{Q}$ of size $n + s$ and of degree s that computes $\mathcal{C}$.*

3.3.3 Pinocchio: a VC protocol from QAPs

In the previous section, we defined QAPs as very efficient objects to encode computations, even non-deterministic ones. QAPs seem a priori very suited to build VC schemes but they cannot be securely used directly. Indeed, in the QAP definition, the divisibility of the built polynomial p by the target polynomial t does not imply the correctness of the circuit evaluation if *different* coefficients are used to build the polynomials v,w and y. Take our running example, providing the values $(30, 13, 3, 12)$ as input gives an output value of 14 and intermediate values $(21, 6)$. A cheating prover can first choose an arbitrary output value and then build v,w and y with the good input values but with intermediate values chosen such that the divisibility check holds. In our example, after choosing 5 as output, setting $(30, 13, 3, 12, 1, 7, 3)$ as coefficients for v, $(30, 13, 3, 12, 0, 0, 3)$ for w and $(30, 13, 3, 12, 21, 6, 3)$ for y leads to a polynomial $p^* = 15 * x^4 + 24 * x^3 + 16 * x + 7$ that is still divisible by t (the correct polynomial p for the same inputs should be $p(x) = 3 * x^4 + 22 * x^3 + 28 * x + 9$). To protect the verifier against a cheating prover, Gennaro et al. [GGPR13] use an homomorphic encryption scheme to force the prover to work over encryptions of the polynomial families and thus to behave correctly. In detail, the prover will encrypt the evaluation of v, w and y at a point s that neither he nor the verifier know. To enable the verifier to check the divisibility, the prover is asked to compute the quotient polynomial $h := p/t$ and to provide $h(s), v(s), w(s), y(s)$. Encryptions of the values $\{v_k(s)\}$, $\{w_k(s)\}$, $\{y_k(s)\}$ and of powers of s define the evaluation key. These values enable the prover to build the values $h(s), v(s), w(s), y(s)$. The verifier, having access to $t(s)$, has therefore to check that:

$$t(s) \cdot h(s) = v(s) \cdot w(s) - y(s) \tag{3.16}$$

The scheme should thus be additively homomorphic to perform the polynomial evaluations over encrypted data and multiplicatively homomorphic in order to perform this last check. Gennaro et al. [GGPR13] propose to use Paillier's encryption [Pai99]. Parno et al. [PHGR13] rather leverage encryption in bilinear groups. Even if the scheme is only additively homomorphic, pairings enable to check relationship involving a product. A new problem arises when using encryption of the polynomials: how can the prover be sure that the encrypted value he receives have been computed correctly ? Additional elements need to be appended to the proof in order to check that the prover indeed computed the values $v(s)$, $w(s)$ and $y(s)$ from the values of the families $\{v_k(s)\}$, $\{w_k(s)\}$,

$\{y_k(s)\}$. Verifier also need to be convinced that the prover used the same linear combination while computing v, w and y. Additional elements in the proof enable the verifier to check that last condition. In the sequel, we will describe the implementation of such VC scheme as done by Parno et al.

The Pinocchio protocol [PHGR13]

Assume that $\mathcal{C}\colon \mathbb{F}^n \to \mathbb{F}^{n'}$ is an arithmetic circuit with d multiplicative gates and define N as $N = n + n'$. By theorem 5, there exists a QAP $\mathcal{Q} = \{\mathcal{V} = (v_k(x)), \mathcal{W} = (w_k(x)), \mathcal{Y} = (y_k(x))\}_{k\in\{0,1,\ldots,m\}}$ of size $m := d + N$ and degree d that computes $\mathcal{C}$. Let denote $\mathbb{G}$ a symmetric bilinear group, g a generator of $\mathbb{G}$ and e the bilinear map associated to $\mathbb{G}$: $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. As the group $\mathbb{G}$ will be a subgroup of an elliptic curve, we use additive notations for elements in $\mathbb{G}$. Like in Parno et al. [PHGR13], the protocol is described in a symmetric bilinear group but the actual implementation leverages asymmetric bilinear groups for efficiency. Some background on symmetric and asymmetric bilinear groups can be found in Section 2.2.3. The Pinocchio protocol is the following:

`KeyGen`

- Pick $r_v, r_w, s, \alpha_v, \alpha_w, \alpha_y, \beta, \gamma \xleftarrow{\$} \mathbb{F}$
- Set $r_y = r_v \cdot r_w$, $g_v = [r_v]g$, $g_w = [r_w]g$ and $g_y = [r_y]g$.
- Denote respectively IO the set of wire labels of the circuit corresponding to inputs and outputs and NIO the wire labels of the intermediate values of the circuit: $IO = \{0, \ldots, N\}$ and $NIO = \{N+1, \ldots, m\}$. Therefore, the public **evaluation key** is:

$$EK = \begin{Bmatrix} \{[v_k(s)] \cdot g_v, [w_k(s)] \cdot g_v, [y_k(s)] \cdot g_v, [\alpha_v v_k(s)] \cdot g_v, [\alpha_w w_k(s)] \cdot g_v, [\alpha_y y_k(s)] \cdot g_v\}_{k\in NIO} \\ \{[\beta v_k(s)] \cdot g_v + [\beta w_k(s)] \cdot g_w + [\beta y_k(s)] \cdot g_y\}_{k\in NIO}, \{[s^i]g\}_{i\in\{0,\ldots,d\}} \end{Bmatrix} \tag{3.17}$$

- The public **verification key** is:

$$VK = \left\{g, [\alpha_v]\cdot g, [\alpha_w]\cdot g, [\alpha_y]\cdot g, [\gamma]\cdot g, [\beta\gamma]\cdot g, [t(s)]\cdot g_y, \{[v_k(s)]\cdot g_v, [w_k(s)]\cdot g_v, [y_k(s)]\cdot g_v\}_{k\in IO}\right\} \tag{3.18}$$

`Prove`

- From input $a = (a_1, \ldots, a_N) \in \mathbb{F}^N$, the prover computes $y = \mathcal{C}(a)$ and gets the intermediate values $(a_{N+1}, \ldots, a_m)$.
- Being in possession of a tuple $(a_1, \ldots, a_m)$, the prover then computes the polynomial p and the quotient polynomial: $h = p/t$ (see definition 13).
- Using the evaluation key EK, the prover builds the proof, which has 8 elements, as:

$$\pi = \begin{Bmatrix} [v_{NIO}(s)] \cdot g_v, [w_{NIO}(s)] \cdot g_w, [y_{NIO}(s)] \cdot g_y, [h(s)] \cdot g, \\ [\alpha_v v_{NIO}(s)] \cdot g_v, [\alpha_w w_{NIO}(s)] \cdot g_w, [\alpha_y y_{NIO}(s)] \cdot g_y, \\ [\beta v_{NIO}(s)] \cdot g_v + [\beta w_{NIO}(s)] \cdot g_w + [\beta y_{NIO}(s)] \cdot g_y \end{Bmatrix} \tag{3.19}$$

where $v_{NIO}(x) = \sum_{i\in NIO} a_i v_i(x)$, $w_{NIO}(x) = \sum_{i\in NIO} a_i w_i(x)$ and $y_{NIO}(x) = \sum_{i\in NIO} a_i y_i(x)$

`Verify`

Upon receiving a claimed proof π, the verifier parses the proof as:

$$\pi = \{[V_{NIO}] \cdot g, [W_{NIO}] \cdot g, [Y_{NIO}] \cdot g, [H] \cdot g, [V'_{NIO}] \cdot g, [W'_{NIO}] \cdot g, [Y'_{NIO}] \cdot g, [Z] \cdot g\} \tag{3.20}$$

and then proceeds as follows:

- leveraging the public verification key, the verifier computes the following values:

$$[v_{IO}(s)] \cdot g_v = \sum_{k \in IO} [a_k] \cdot ([v_k(s)] \cdot g_v) \tag{3.21}$$

$$[w_{IO}(s)] \cdot g_w = \sum_{k \in IO} [a_k] \cdot ([w_k(s)] \cdot g_w) \tag{3.22}$$

$$[y_{IO}(s)] \cdot g_y = \sum_{k \in IO} [a_k] \cdot ([y_k(s)] \cdot g_y) \tag{3.23}$$

- using the pairing e, the verifier first performs the divisibility check:

$$\begin{aligned} &e([v_0(s)] \cdot g_v + [v_{IO}(s)] \cdot g_v + [V_{NIO}] \cdot g, [w_0(s)] \cdot g_w + [w_{IO}(s)] \cdot g_w + [W_{NIO}] \cdot g) \\ &= e([t(s)] \cdot g_y, [H] \cdot g) \times e([y_0(s)] \cdot g_y + [y_{IO}(s)] \cdot g_y + [Y_{NIO}] \cdot g, g) \end{aligned} \tag{3.24}$$

  Then he verifies that the values sent by the prover were indeed computed as linear combination of the v_k, w_k and y_k polynomials:

$$\begin{aligned} e([V'_{NIO}] \cdot g, g) &= e([V_{NIO}] \cdot g, [\alpha_v] \cdot g) \\ e([W'_{NIO}] \cdot g, g) &= e([W_{NIO}] \cdot g, [\alpha_w] \cdot g) \\ e([Y'_{NIO}] \cdot g, g) &= e([Y_{NIO}] \cdot g, [\alpha_y] \cdot g) \end{aligned} \tag{3.25}$$

  Finally, the verifier checks that, in the calculation of the polynomial p, the prover provided the same coefficient for the linear combinations involving the v_k, w_k and y_k polynomials:

$$e([Z] \cdot g, [\gamma] \cdot g) = e([V_{NIO}] \cdot g + [W_{NIO}] \cdot g + [Y_{NIO}] \cdot g, [\beta\gamma] \cdot g) \tag{3.26}$$

Making the proof zero-knowledge

In a genuine proof, the correctness of a computation is checked via the divibility check performed in (3.24), which verifies if the polynomial p computed from the circuit is indeed divisible by the target polynomial t. Adding a random multiple to each of the proof elements does not change the divisibility by t but it has the effect of randomizing the proof. Gennaro et al. [GGPR13] showed that such proof is zero-knowledge, in the sense that the prover can provide inputs that will not be revealed in the proof. Parno et al. [PHGR13] show how to modify the evaluation keys to compute this zero-knowledge proof: they add multiples of $t(s)$ to help the prover in building the randomized proof. Compared to a plain proof, the computation of the zero-knowledge proof requires few additional operations. Such zero-knowledge proofs are examples of zk-SNARKs (see Section 2.1.4): the succinctness is achieved because the proof has constant size.

Security of Pinocchio

The security of Pinocchio relies on the q-power knowledge of exponent (q-PKE), the q-power Diffie-Hellman (q-PDH assumptions) and the q-strong Diffie-Hellman (q-SDH). The q-power knowledge of exponent and q-power Diffie-Hellman assumptions have been defined by Groth [Gro10] and the q-strong Diffie-Hellman assumption by Boneh and Boyen [BB04]. The q-PDH assumption roughly states that even if an adversary knows a sequence of powers of a secret value s encrypted in points of $\mathbb{G}$ except one of the power, it can recover the missing one only with negligible probability. The q-SDH assumption states that an adversary seeing consecutive powers of a secret value s encrypted in points of $\mathbb{G}$ cannot build a value in the target group $\mathbb{G}_T$ that depends on s. Finally, the q-PKE assumption states that if an adversary sees consecutive powers of s and consecutive powers of αs for a randomly chosen secret α, it is impossible to compute a power of αs without knowing the value α.

The intuition behind these assumption is the following:

- If the q-PKE assumption holds then the adversary must have build the terms $[v_{NIO}(s)] \cdot g_v$ and $[\alpha_v v_{NIO}(s)] \cdot g_v$ by respectively computing a linear combination of the terms $[v_k(s)] \cdot g_v$ and of $[\alpha_v v_k(s)] \cdot g_v$.
- If the q-PDH assumption holds then the adversary must have used the same linear combination while computing the polynomials v_{NIO}, w_{NIO} and y_{NIO}.
- If the 2q-SDH assumption holds then the adversary must have computed an encrypted value of the polynomial $h(s)$ that is indeed the result of the division of the polynomial $p(x)$ by $t(x)$ evaluated at s.

Parno et al. [PHGR13] show that if the QAP computed from the circuit to verify has d multiplicative gates, their protocol is sound under the d-PKE, (4d+4)-PDH and (8d+8)-SDH assumptions.

Pinocchio and SNARKs

Bitansky et al. defined the notion of Succinct Non-Interactive Argument of Knowledge [BCCT12] (SNARK) in [BCCT12]. The difference with non-interactive arguments resides in the length of the resulting argument: SNARK proofs for non-deterministic computations are *independent* of the size of the witness. Groth [Gro10] gave the first argument with *constant* size at the expense of a long common reference string (CRS), whose size was later decreased by Lipmaa [Lip12]. Groth's arguments have 42 group elements. Gennaro et al.'s SNARK construction [GGPR13] enables to build an argument with only 9 group elements. Parno et al.'s Pinocchio [PHGR13] gives an argument with 8 group elements.

Let $\mathcal{C}$ be a **NP**-statement, expressed as an arithmetic circuit over a finite field $\mathbb{F}$ and λ be a security parameter. We denote by w a witness of a valid statement. A zk-SNARK scheme is defined by three polynomial-time algorithms ($\texttt{KeyGen}, \texttt{Prove}, \texttt{Verifiy}$), such that:

- $(ek_{\mathcal{C}}, vk_{\mathcal{C}}) \leftarrow \texttt{KeyGen}(1^{\lambda}, \mathcal{C})$: the randomized `KeyGen` algorithm takes as input a security parameter and an arithmetic circuit and produces two public keys, an evaluation key $ek_{\mathcal{C}}$ and a verification key $vk_{\mathcal{C}}$.

- $(y, \pi) \leftarrow \texttt{Prove}(ek_{\mathcal{C}}, x, w)$: the deterministic `Prove` algorithm, takes as inputs a value x, a witness w and the evaluation key $ek_{\mathcal{C}}$ and outputs $y = \mathcal{C}(x, w)$ along with a proof π that y has been correctly computed.

- $\{0, 1\} \leftarrow \texttt{Verify}(vk_{\mathcal{C}}, x, y, \pi)$: the deterministic `Verify` algorithm takes the input/output (x, y) of the circuit $\mathcal{C}$, the proof π and the verification key $vk_{\mathcal{C}}$ and outputs 1 if there exists a witness w such that $y = \mathcal{C}(x, w)$ and 0 otherwise.

A zk-SNARK reaches the following security properties that we first describe informally.

- *Completeness*: if there exists a satisfying witness for the statement $\mathcal{C}$, the verifier should always accept a proof produced by an honest prover.

- *Proof of knowledge*: if the proof of a statement $\mathcal{C}$ is accepted by the verifier, it means that not only a witness exists for the statement but also that the prover indeed knows this witness. This is formalized with an efficient algorithm that is able to extract the witness from the proof.

- *Zero-knowledge*: the proof reveals no more information about the witness that what could be inferred from the result of the computation. This is formalized with an algorithm that can simulate proofs, zero-knowledge is thus reached if a verifier cannot distinguish between a proof produced by the prover and a proof produced by the simulator.

- *Efficiency*: the proof has a polynomial size *in the security parameter* and the verifier algorithm runs in time that is polynomial in the security parameter and the input length.

3.3.4 zk-SNARK formal definition

We use Ben-Sasson et al. [BCG+13] definitions of zk-SNARKs.

Definition 14. *A triple of algorithms* $(\texttt{KeyGen}, \texttt{Prove}, \texttt{Verify})$ *is a publicly verifiable zk-SNARK if the following conditions are satisfied.*

1. Completeness*: for every large enough security parameter* λ*, every circuit* $\mathcal{C} : \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$*, every input* $x \in \{0,1\}^n$ *and every assignment* $a \in \{0,1\}^h$ *with* $(x, a) \in \mathcal{R}_{\mathcal{C}}$*,*

$$Pr\left[\texttt{Verify}(vk_{\mathcal{C}}, x, \pi) \;\middle|\; \begin{array}{rl} (ek_{\mathcal{C}}, vk_{\mathcal{C}}) & \leftarrow \texttt{KeyGen}(1^\lambda, \mathcal{C}) \\ \pi & \leftarrow \texttt{Prove}(ek_{\mathcal{C}}, x, a) \end{array}\right] = 1$$

2. Proof of knowledge*: for every polynomial size prover* P^* *there exists a polynomial size extractor* E *such that for every constant* $c > 0$*, large enough security parameter* $\lambda \in \mathbb{N}$*, every auxiliary input* $z \in \{0,1\}^{\text{poly}(\lambda)}$ *and every circuit* $\mathcal{C} : \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ *of size* λ^c*,*

$$Pr\left[\begin{array}{r} \texttt{Verify}(ek_{\mathcal{C}}, x, \pi) = 1 \\ (x, a) \notin \mathcal{R} \end{array} \;\middle|\; \begin{array}{rl} (ek_{\mathcal{C}}, vk_{\mathcal{C}}) & \leftarrow \texttt{KeyGen}(1^\lambda, \mathcal{C}) \\ (x, \pi) & \leftarrow P^*(z, ek_{\mathcal{C}}, vk_{\mathcal{C}}) \\ a & \leftarrow E(z, ek_{\mathcal{C}}, vk_{\mathcal{C}}) \end{array}\right] \leqslant \text{negl}(\lambda)$$

3. Zero knowledge*: there exists a stateful interactive polynomial size simulator $\mathcal{S}$ such that for all stateful interactive polynomial size distinguisher $\mathcal{D}$, constant c, large enough security parameter $\lambda \in \mathbb{N}$, every auxiliary input $z \in \{0,1\}^{\text{poly}(\lambda)}$ and every circuit $\mathcal{C} : \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ of size λ^c, such that the two distributions* (3.27) *and* (3.28) *above are statistically close:*

$$Pr\left[\begin{array}{c|l} (x,a) \in \mathcal{R} \\ D(\pi) = 1 \end{array} \middle| \begin{array}{rl} (ek_{\mathcal{C}}, vk_{\mathcal{C}}) & \leftarrow \texttt{KeyGen}(1^\lambda, \mathcal{C}) \\ (x, \pi) & \leftarrow D(z, ek_{\mathcal{C}}, vk_{\mathcal{C}}) \\ \pi & \leftarrow \texttt{Prove}(ek_{\mathcal{C}}, x, a) \end{array}\right] \tag{3.27}$$

$$Pr\left[\begin{array}{c|l} (x,a) \in \mathcal{R} \\ D(\pi) = 1 \end{array} \middle| \begin{array}{rl} (ek_{\mathcal{C}}, vk_{\mathcal{C}}, \text{trap}) & \leftarrow \mathcal{S}(1^\lambda, \mathcal{C}) \\ (x, a) & \leftarrow \mathcal{D}(z, ek_{\mathcal{C}}, vk_{\mathcal{C}}) \\ \pi & \leftarrow \mathcal{S}(z, ek_{\mathcal{C}}, x, \text{trap}) \end{array}\right] \tag{3.28}$$

4. Efficiency*: there is a universal polynomial p such that for every large enough security parameter $\lambda \in \mathbb{N}$, every circuit $\mathcal{C} : \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, input $x \in \{0,1\}^n$ and assignment $a \in \{0,1\}^h$ with $(x,a) \in \mathcal{R}_{\mathcal{C}}$,*

   - *the verifying algorithm runs in time $p(\lambda + |x|)$*
   - *an honestly generated proof has size $p(\lambda)$.*

3.3.5 Groth's zk-SNARK (Groth16)

Bitansky et al. [BCI+13] noticed that Gennaro et al.'s QAP construction [GGPR13] did not follow the framework often used to design arguments. Usually, designers start from an information-theoretic proof system, such as e.g. a PCP, and restrict the possibility of the prover by the mean of cryptographics tool applied to the information-theoretic proof system. They came up with an information-theoretic framework called *linear-interactive proofs* that can then be compiled into a SNARK, provided that the prover of the information-theoretic framework is restricted to compute linear functions of the messages exchanged with the verifier during the protocol. This framework is compiled into an argument by the mean of an homomorphic encryption scheme that restricts the prover to perform linear operations on the ciphertexts. The verifier can then encrypt queries and send them to the prover that can only perform linear operations on the query due to the encryption scheme. Bitansky et al. show that Gennaro et al.'s SNARK construction [GGPR13] fits into the linear interactive proof framework: the QAP construction induces a linear interactive proof and the encryption scheme (in the bilinear group) is homomorphic. The location of the queries to the linear interactive proof are encrypted and stored in the common reference string (CRS) of the scheme. They correspond to the encryption of the QAP evaluations at a random point, denoted by s above. Since the prover does not know the value s and cannot recover it from the encrypted values in the CRS, he is only able to compute linear combinations of the encrypted values to compute the proof. Note that the prover can also be restricted to perform affine operations on the messages if the encryption of the constant 1 is added in the common reference string.

Following Bitansky et al. [BCI+13], Groth [Gro16] proposed the most efficient QAP-based SNARK: the proof only consists of 3 group elements. Groth also proves that in the linear interactive proof framework, it is impossible to build a SNARK that has only one group for the proof. Groth's construction is thus quasi-optimal since there is only place

left for a SNARK that has 2 group elements. We describe Groth's SNARK construction in the sequel. It takes place in an asymmetric bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g)$ where:

- $\mathbb{G}_1$,$\mathbb{G}_2$ and $\mathbb{G}_T$ have order p, a prime number,
- The pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map,
- g is a generator of $\mathbb{G}_1$, h a generator of $\mathbb{G}_2$ and $e(g,h)$ a generator of $\mathbb{G}_T$.

To simplify the notations, elements in the groups are represented by their logarithms: $[\alpha]_1$ stands for $[\alpha]\cdot g$, $[\alpha]_2$ stands for $[\alpha]\cdot h$ while $[\alpha]_T$ stands for $e(g,h)^\alpha$. Note that the pairing enables to write 'products' between elements of $\mathbb{G}_1$ and $\mathbb{G}_2$, the result belongs to the target group $\mathbb{G}_T$. Writing $[\alpha]_1 \cdot [\beta]_2 = [\alpha\cdot\beta]_T$ makes sense since:
$e([\alpha]_1, [\beta]_2) = e([\alpha]\cdot g, [\beta]\cdot h) = e(g,h)^{\alpha\beta} = [\alpha\beta]_T$.

`KeyGen`

- Pick $\alpha, \beta, \gamma, \delta, s \xleftarrow{\$} \mathbb{F}_p^*$.
- Define $\tau = (\alpha, \beta, \gamma, \delta, s)$
- Compute $\sigma = (\sigma_P, \sigma_V)$, where:

$$\sigma_P = \left(\begin{array}{l} [\alpha]_1, [\beta]_1, [\beta]_2, [\delta]_1, [\delta]_2, [\gamma]_2, \{[s^i]_1\}_{i=0}^{n-1}, \{[s^i]_2\}_{i=0}^{n-1}, \left\{\left[\dfrac{\beta u_i(s) + \alpha v_i(s) + w_i(s)}{\gamma}\right]_1\right\}_{i=0}^{\ell}, \\ \left\{\left[\dfrac{\beta u_i(s) + \alpha v_i(s) + w_i(s)}{\delta}\right]_1\right\}_{i=\ell+1}^{m}, \left\{\left[\dfrac{s^i t(s)}{\delta}\right]\right\}_{i=0}^{n-2} \end{array} \right) \tag{3.29}$$

$$\sigma_V = \left([1]_1, [\alpha]_1, [1]_2, [\alpha]_2, [\gamma]_2, [\delta]_2, \left\{\left[\frac{\beta u_i(s) + \alpha v_i(s) + w_i(s)}{\gamma}\right]_1\right\}_{i=0}^{\ell} \right) \tag{3.30}$$

`Prove`$(\sigma, a_1, \ldots, a_m)$

- Pick $r_A, r_B \xleftarrow{\$} \mathbb{F}_p$
- Compute:

$$[A]_1 = \left[\alpha + \sum_{i=0}^{m} a_i u_i(s) + r_A\delta\right]_1 \tag{3.31}$$

$$[B]_2 = \left[\beta + \sum_{i=0}^{m} a_i v_i(s) + r_B\delta\right]_2 \tag{3.32}$$

$$[C]_1 = \left[\sum_{i=0}^{m} \frac{a_i(\beta u_i(s) + \alpha v_i(s) + w_i(s))}{\delta} + \frac{h(s)t(s)}{\delta} + Ar_{\mathrm{B}} + Br_{\mathrm{A}} + r_{\mathrm{A}}r_B\,\delta\right]_1 \tag{3.33}$$

- The proof is $\pi = ([A]_1, [B]_2, [C]_1)$.

`Verify`$(\sigma, \pi, a_1, \ldots, a_\ell))$

- Parse π as $([A]_1, [B]_2, [C]_1) \in \mathbb{G}_1^2 \times \mathbb{G}_2$.

- Accept π if and only if:

$$e([A]_1, [B]_2) = e([\alpha]_1, [\beta]_2) + e\left(\sum_{i=0}^{\ell} a_i \left[\frac{\beta u_i(s) + \alpha v_i(s) + w_i(s)}{\gamma}\right], \right) + e([C]_1, [\delta]_2) \tag{3.34}$$

The verifier performance can be improved by replacing the two values $[\alpha]_1$ and $[\beta]_2$ in the verification key σ_V by the value $[\alpha\beta]_T$. This replacement removes a pairing operation for the verifier: instead of computing $e([\alpha]_1, [\beta]_2)$ the verifier has already access to this value because it is equal to $[\alpha\beta]_T$. Note that the proof is zero-knowledge by design since the three elements of the proof are randomized: $[A]_1$ contains $r_A\delta$, $[B]_2$ contains $r_B\delta$ and $[C]_1$ contains $r_A r_B\delta$.

Security

In the linear interactive proof (LIP) framework, once the information-theoretic proof scheme is defined there are different ways to compile the proof scheme into an argument. Groth chooses to compile its LIP that contains 3 group elements into an argument that also contains 3 group elements. This 'aggressive' compilation comes at the expanse of a stronger cryptographic assumption, namely the *generic group model* [Sho97]. In this model, it is assumed that an adversary only has a black-box access to group operations of the curve. For instance, to add two points on the curve, the adversary has to query an addition oracle by sending the two points he wants to add. The oracle then sends back the adversary the sum of the two given points.

3.3.6 A remark on the setup phase

Parno et al.'s protocol and Groth's protocol both require a setup phase (the `KeyGen` algorithm). The phase generates secret values, notably the value s where the polynomials will be evaluated. The `KeyGen` algorithm must be run by a trusted party because anyone in possession of the secret values generated during the setup can forge proofs of incorrect statements that will nonetheless be accepted. Campanelli et al. [CGGN17] build and implement attacks on a zk-SNARK protocol by first tampering the `KeyGen` phase. They are then able to break the zero-knowledge property of the scheme.

To ensure that the setup phase was run properly, Ben-Sasson et al. [BCG+15] design a secure multiparty computation protocol to securely run the `KeyGen` protocol as long as at least one participant of the protocol is honest.

Another line of work builds on the notion of *subversion-resistant* protocols [BFS16] to design a different `KeyGen` algorithm where the resulting evaluation and verification keys consistency can be checked [Fuc18, GKM+18].

3.4 Highlighting the Gaps

The objective of our study is general-purpose verifiable computation aiming at practicality. Therefore, one of the main goals is to increase the verifier's efficiency *without* sacrificing the prover's. Table 3.2 summarizes the complexity of verifiers and provers in the schemes described in the chapter while table 3.3 sums up the expressiveness of such schemes. As mentioned in Section 3.1, interactive proofs yield very efficient VC

	Prover	Verifier	Bandwidth / Proof size
Interactive Proofs	linear	sub-linear	sub-linear
Interactive Arguments	linear	sub-linear	sub-linear
Non-interactive Arguments	linear (crypto ops) quasi-linear (non-crypto ops)	linear in the *input size*	constant

Table 3.2 – Performance of state of the art VC schemes (except when mentioned, complexities depend on the circuit size).

protocols: for some class of computations, the state of the art protocols achieve complexity linear in the circuit size for the prover and sublinear complexity for the verifier, which reaches the goal mentioned above. However these protocols are not fully general-purpose: in order to be efficient the computation to verify must be structured like a parallel computation with the same subcomputation repeated several times, e.g. at least ten times in [WTS+18] for the scheme to reach efficiency for the verifier.

More general computations, notably non-deterministic computations, can be verified using arguments as described in Section 3.2. However, these protocols are built from interactive proofs and, even though they allow for interesting properties and help preserve the prover's efficiency, their drawback is the requirement for interaction between the verifier and the prover to establish the argument and the resulting communication overhead. Moreover, the expressiveness in terms of the variety of computations that can be verified is not fully achieved: there still are some class of computations that do not lend themselves to efficient verification.

Non-interactive verifiable computation schemes leveraging QAP-based non-interactive arguments achieve several interesting properties: they can deal with virtually every kind of computation, the proof has constant and short size regardless of the complexity of the computation to verify and therefore the verification is very efficient. Moreover, such QAP-based schemes have additional properties like public verifiability or perfect zero-knowledge for the proof. Such schemes seem to perfectly fit our requirement for general-purpose VC schemes, however the drawback is the large computation overhead for the prover: for a circuit with N multiplication gates, the complexity of the state of the art prover is $O(N \log N)$ non-cryptographic work and $O(N)$ cryptographic work: there is at least one public-key cryptographic operation for each multiplication gate to produce the proof. Compared to the interactive prover designed by Thaler [Tha13] that has a $O(N)$ complexity, with no public-key cryptographic operations, the difference in efficiency is huge. Therefore, there is room for improvement of non-interactive prover's efficiency. This raises the following question:
Is it possible to improve the efficiency of the prover in VC systems while keeping a general-purpose protocol and the succinctness of the proof ?
Ideally, such VC system should achieve the following requirements:

- It should be *expressive*: the VC system should be able to verify any **NP**-problem. A VC system able to deal with a large subset of a high-level programming language would meet such a requirement.
- The entities involved in the VC scheme should be *efficient*: ideally the prover should not have much more work than executing the circuit that implements the

	Expressiveness	
	Auxiliary inputs support	Circuit types
Interactive proofs	No	Low-depth circuits
Interactive Arguments	Yes	Low-depth circuits
Non-interactive Arguments	Yes	All

Table 3.3 – Expressiveness of state of the art VC schemes.

function to verify, while the verifier should not have more work than reading the input of the outsourced computation to verify the correctness of the computation. As a consequence, proofs produced by the prover should be short (constant size or at least polylogarithmic in the input size).

- With practicality in mind, the VC system should not only have good asymptotics for the prover, the verifier and the proof size as required above but it should also exhibit good hidden constants in these asymptotics such that the implementation of the scheme in a prototype would be efficient.

In the sequel, we propose to leverage proof composition to define a VC scheme approaching the above goals: we embed efficient but specialized protocols into a state-of-the-art non-interactive scheme. The resulting scheme preserves the proof succinctness and a high degree of expressiveness while increasing the efficiency of the prover.

Chapter 4

Proof Composition

4.1 Motivation: increase prover's efficiency in machine learning algorithms

While achieving excellent results in diverse areas, machine learning algorithms require expertise and a large training material to be fine-tuned. Outsourcing machine learning algorithms helps users to deal with large amounts of data without the need to develop the expertise required by these algorithms. Therefore, cloud providers such as Amazon or Microsoft have started offering Machine Learning as a Service (MLaaS) to perform complex machine learning tasks. Outsourcing however raises severe security issues due to potentially untrusted service providers and raises a new requirement: in the face of potentially malicious service providers, the users need additional guarantees to gain confidence in the results of outsourced computations. Verifiable computing (VC) tackles some of these issues by providing computational integrity for an outsourced computation and provides proofs of computational integrity without any assumptions on hardware or on potential failures. Existing VC systems can theoretically prove and verify all **NP** computations (see chapter 3 for details), nevertheless, despite the variety of existing solutions, existing VC schemes have to make trade-offs between expressiveness and functionality, see Section 3.4. Therefore, they cannot efficiently handle the verifiability of a sequence of operations with a high variance in nature and complexity, like the ones involved in machine learning techniques. Even if expressive VC schemes such as Pinocchio [PHGR13] can ensure the verifiability of a machine learning algorithm, the cryptographic work required to produce the proof prevents from dealing with large but simple computations such as matrix multiplications. On the other hand, some schemes like Cormode et al.'s CMT [CMT12] are very efficient and can deal with large computations, e.g. large matrix multiplications, but cannot handle the variety of even very simple operations such as integer comparisons. Hence there is a need for a VC scheme that achieves both efficiency by handling complex operations and expressiveness through the variety of types of operations it can support. In this chapter, we propose a scheme that combines a general purpose VC scheme like Pinocchio and various specialized VC schemes that achieve efficient verification of complex operations like large matrix multiplications. We denote by **GVC** the general purpose VC scheme and **EVC** the specialized VC schemes.

Existing works have already considered the problem of outsourcing machine learning

algorithms: in SafetyNets [GGG17], Ghodsi et al. build an interactive proof protocol to verify the execution of a deep neural network on an untrusted cloud. This approach, albeit efficient, has several disadvantages over ours. The first is that expressivity of the interactive proof protocol used in SafetyNets prevents using state of the art activation functions such as ReLU. Following CryptoNets [GDL+16], Ghodsi et al. replace ReLU functions by a quadratic activation function, namely $x \mapsto x^2$, which squares the input values element-wise. The fact that square activation functions have unbounded derivative causes instability on the neural network training, as mentioned in [GDL+16]. A second disadvantage is the impossibility for the prover to hide some of his inputs, i.e. to prove a non-deterministic computation (see Section 3.1.2). As a consequence, the verifier and the prover of SafetyNets have to share the neural network model, namely the values of the matrices that represent the linear operations of the neural network. This situation is quite unusual in machine learning: since the training of neural networks is expensive and requires a large amount of data, powerful hardware and technical skills to obtain a classifier with good accuracy, it is unlikely that cloud providers share their models with users. In contrast, with our proposed method the prover could keep the model private and nonetheless be able to produce a proof of correct execution.

4.2 State of the Art in Proof Composition

4.2.1 Ben Sasson et al.'s Recursive Composition of zk-SNARKs

Building on Valiant's incrementally verifiable computations [Val08], Bitansky et al. [BCCT13] develop recursive composition for SNARKs by proving there exists a SNARK that produces a proof that i) a computation has been carried out correctly, ii) a previous SNARK proof has passed the verification. Albeit proving the existence of such recursive composition under reasonable assumptions, Bitansky et al.'s work is essentially theoretical.

In [BCTV14a], Ben Sasson et al. take Bitansky et al.'s theoretical work and tackle several challenges to obtain a concrete instantiation of recursive SNARKs. If efficient, the resulting systems would allow to divide a complex program to verify into several smaller parts, to produce a proof for each sub-part and, via recursive composition, to compress the proofs into a single short proof. The high level idea of the Ben-Sasson et al.'s proof system is to prove or verify the satisfiability of an arithmetic circuit that checks the validity of the previous proofs. Thus, the verifier should be implemented as an arithmetic circuit and used as a sub-circuit of the next prover. However, SNARKs verifiers perform the verification checks using an elliptic curve pairing and it is mathematically impossible for the base field to have the same size as the elliptic curve group order. Ben-Sasson et al. therefore propose a cycle of elliptic curves to enable proof composition. When two such elliptic curves form a cycle, the finite field defined by the prime divisor in the group order of the first curve is equal to the base field (or field of definition) of the second curve and vice versa. Although proofs can theoretically be composed as many times as desired, this method has severe overhead, as shown by experimental results in the paper. The method we propose in this chapter has a more limited spectrum than Ben-Sasson et al.'s but our resulting system is still general purpose and enjoys the property of the GVC system, such as succinctness or efficiency for the prover. Furthermore, our proposal improves the prover time, replacing a part of a computation by sub-circuit verifying the sub-computation that can then be executed

outside the prover.

4.2.2 Costello et al.'s Geppetto

Building on Pinocchio [PHGR13], Costello et al. design a proof system called Geppetto [CFH+15] that can deal with computations decomposed into smaller sub-computations that share values. Their system produces as many proofs as there are sub-computations. They further adapt the recursive composition proposed by Ben Sasson et al. [BCTV14a] to end with a single proof for the whole computation: noticing that, although Ben Sasson et al. technique enables to compose an unbounded number of proofs, their recursive composition technique is way inefficient, Costello et al. limit the number of proof that can be composed. Therefore, they are able to choose more efficient and secure (128 bit instead of 80 bit of security) elliptic curves to embed the verification algorithm of the sub-proofs. Compared to our proposal, the drawback remains similar since the outer elliptic curve still has to verify a pairing computation. Besides, Geppetto also leverage commitments for computations that share states and the verification of these commitments also incurs overheads. Finally, even if Geppetto increases the expressiveness of Pinocchio and obtains an efficient verifier with proof composition, the efficiency of the prover is not improved.

4.3 Embedded Proofs

4.3.1 Problem Statement

Most applications involve several sequences of function evaluations combined through control structures. Assuring the verifiability of these applications has to face the challenge that the functions evaluated as part of these applications may feature computational characteristics that are too variant to be efficiently addressed by a unique VC scheme. For instance, in the case of an application that involves a combination of computationally intensive linear operations with simple non-linear ones, none of the existing VC techniques would be suitable since there is no single VC approach that can efficiently handle both. This question is perfectly illustrated by the sample scenario described in the previous section, namely dealing with the verifiability of Neural Network Algorithms, which can be viewed as a repeated sequence of a matrix product and a non-linear activation function. For instance, a two layer neural network, denoted by g, on an input x can be written as:

$$g(x) = W_2 \cdot f(W_1 \cdot x) \tag{4.1}$$

Here W_1 and W_2 are matrices and f is a non-linear function like the frequently chosen Rectified Linear Unit (ReLU) function $x \mapsto max(0, x)$. For efficiency, the inputs are often batched and the linear operations involved in the Neural Network are matrix products instead of products between a vector and a matrix. The batched version of (4.1) therefore is:

$$g(X) = W_2 \cdot f(W_1 \cdot X) \tag{4.2}$$

where X is a batch of inputs to be classified.

In an attempt to assure the verifiability of this neural network, two alternative VC schemes seem potentially suited: the CMT protocol [CMT12] based on interactive proofs and Pinocchio [PHGR13]. CMT can efficiently deal with the matrix products but problems arise when it comes to the non-linear part of the operations since, using CMT,

each function to be verified has to be represented as a layered arithmetic circuit (i.e. as an acyclic graph of computation over a finite field with an addition or a multiplication at each node, and where the circuit can be decomposed into layers, each gate of one layer being only connected to an adjacent layer). Nevertheless the second component of the neural network algorithm, that is, the ReLU activation function, does not lend itself to a simple representation as a layered circuit. [GKR08] and [CMT12] have proposed solutions to deal with non-layered circuits at the cost of very complex pre-processing resulting in a substantial increase in the prover's work and the overall circuit size. Conversely, Pinocchio eliminates the latter problem by allowing for efficient verification of the non-linear ReLU activation function while suffering from excessive complexity in the generation of proofs for the products of large matrices (benchmarks on matrix multiplication proofs can be found in [WB15]).

This sample scenario points to the basic limitation of existing VC schemes in efficiently addressing the requirements of common scenarios involving several components with divergent characteristics such as the mix of linear and non-linear operations as part of the same application. The objective of our work therefore is to come up with a new VC scheme that can efficiently handle these divergent characteristics in the sub-components as part of a single VC protocol. We propose a new proof composition scheme where the resulting VC scheme:

1. efficiently addresses the verifiability of a sequence of operations,

2. inherits the properties of the outer scheme, namely Pinocchio, and can thus provide privacy for inputs supplied by the prover.

4.3.2 Idea of the Solution: Embedded Proofs

Our solution is based on a method that enables the composition of a general purpose VC scheme suited to handle sequences of functions with one or several specialized VC schemes that can achieve efficiency in case of a component function with excessive requirements like very large linear operations. Without loss of generality, we apply the method to a pair of VC schemes, assuming that one is a general purpose VC scheme, called `GVC`, like Pinocchio [PHGR13], which can efficiently assure the verifiability of an application consisting of a sequence of functions, whereas the other VC scheme assures the verifiability of a single function in a very efficient way, like, for instance, a VC scheme that can handle large matrix products efficiently. We call this scheme `EVC`. The main idea underlying the VC composition method is that the verifiability of the complex operation (for which the `GVC` is not efficient) is *outsourced* to the `EVC` whereas the remaining non-complex functions are all handled by the `GVC`. In order to get the verifiability of the entire application by the `GVC`, instead of including the complex operation as part of the sequence of functions handled by the `GVC`, this operation is separately handled by the `EVC` that generates a standalone verifiability proof for that operation and the verification of that proof is viewed as an additional function embedded in the sequence of functions handled by the `GVC`. Even though the verifiability of the complex operation by the `GVC` is not feasible due to its complexity, the verifiability of the proof on this operation is feasible by the basic principle of VC, that is, because the proof is much less complex than the operation itself.

We illustrate the VC composition method using as a running example the Neural Network defined with formula (4.2) in Section 4.3.1. Here, the application consists of

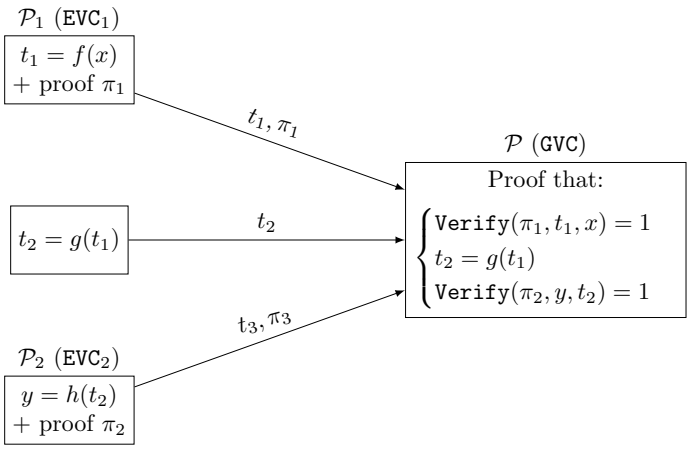


Figure 4.1 – High level view of the embedded proofs

the sequential execution of three functions f, g and h (see Figure 4.1), where f and h are not suitable to be efficiently proved correct by GVC while g is. Note that we consider that g cannot be proved correct by any EVC systems or at least not as efficiently as with the GVC system. The computation to verify is therefore $y = h(g(f(x)))$. In our example, the functions f, g and h are $f : X \mapsto W_1 \cdot X$, $h : X \mapsto W_2 \cdot X$ and $g : X \mapsto \max(0, X)$, where X, W_1 and W_2 are matrices and g applies the max function element-wise to the input matrix X.

In order to cope with the increased complexity of f and h, we have recourse to $\texttt{EVC}_1$ and $\texttt{EVC}_2$ that are specialized schemes yielding efficient proofs with such functions. $\pi_{\texttt{EVC}_1}$ denotes the proof generated by $\texttt{EVC}_1$ on f, $\pi_{\texttt{EVC}_2}$ denotes the proof generated by $\texttt{EVC}_2$ on h and $\Pi_{\texttt{GVC}}$ denotes the proof generated by GVC. For the sequential execution of functions f, g and h, denoting $t_1 = f(x)$ and $t_2 = g(t_1)$, the final proof then is:

$$\Pi_{\texttt{GVC}}\Big(\Big(\texttt{Verif}_{\texttt{EVC}_1}(\pi_{\texttt{EVC}_1}, x, t_1) \stackrel{?}{=} 1\Big) \wedge \Big(g(t_1) \stackrel{?}{=} t_2\Big) \wedge \Big(\texttt{Verif}_{\texttt{EVC}_2}(\pi_{\texttt{EVC}_2}, t_2, y) \stackrel{?}{=} 1\Big)\Big). \quad (4.3)$$

Here the GVC system verifies the computation of g and the verification algorithms of the $\texttt{EVC}_1$ and $\texttt{EVC}_2$ systems, which output 1 if the proof is accepted and 0 otherwise. We note that this method can easily be extended to applications involving more than three functions, Section 4.4 describes the embedded proof protocol for an arbitrary number of functions. Interestingly, various specialized VC techniques can be selected as EVC based on their suitability to the special functions requirements provided that:

1. The verification algorithm for each EVC proof is compatible with the GVC scheme.

2. The verification algorithm for each EVC proof should have much lower complexity than the outsourced computations (by the basic VC advantage).

3. The EVC schemes should not be VC's with a designated verifier but instead publicly verifiable [GGP10]. Indeed, since the prover of the whole computation is the verifier of the EVC, no secret value should be shared between the prover of the EVC and the prover of the GVC. Otherwise, a malicious prover can easily forge a proof for EVC and break the security of the scheme.

In the sequel, we present a concrete instance of our VC composition method using a QAP-based VC scheme as the GVC and an efficient interactive proof protocol, namely the Sum-Check protocol [LFKN90] as the EVC. We further develop this instance with a Neural Network verification example. We first introduce the building blocks required to instantiate our method in Section 4.3.3. Following our embedded proof protocol, we describe a VC scheme involving composition in Section 4.4 and then specialize the GVC and EVC schemes to fit the Neural Network use-case in section 4.5. We compute the related costs of our algorithm in Section 4.6 and report experimental results on the implementation of our scheme in Section 4.7. We finally prove the security of our scheme in Section 4.8 and conclude in Section 4.9.

4.3.3 Building Blocks: Ajtai Hash Function

As mentioned in Section 4.3.1, our goal is to compute a proof of an expensive sub-computation with the Sum-Check protocol and to verify that proof using the Pinocchio protocol. The non-interactive nature of Pinocchio prevents from proving the sub-computation with an interactive protocol. As explained in Section 3.1.1, we turn the Sum-Check protocol into a non-interactive argument using the Fiat-Shamir transform [FS86]. This transformation needs a hash function to simulate the challenges that would have been provided by the verifier. The choice of the hash function to compute challenges in the Fiat-Shamir transformation here is crucial because we want to verify the proof transcript inside the GVC system, which will be instantiated with the Pinocchio protocol. This means that the computations of the hash function have to be verified by the GVC system and that the verification should not be more complex than the execution of the original algorithm inside the GVC system. For instance the costs using a standard hash function such as SHA256 [Nat15] would be too high: [BCG$^+$14] reports about 27,000 multiplicative gates to implement the compression function of SHA256. Instead, we choose a hash function which is better suited for arithmetic circuits, namely the Ajtai hash function [Ajt96] that is described in Section 2.2.2. Few gates are needed to implement an arithmetic circuit for this hash function since it involves multiplications by constants (the matrix A is public): to hash m bits, m multiplicative gates are needed to ensure that the input vector is binary and 3 more gates are needed to ensure that the output is the linear combination of the input and the matrix. With the parameters selected by Kosba et al. in [KZM$^+$15], this means that 1527 gates are needed to hash 1524 bits.

4.4 Embedded Proofs

4.4.1 High level description of the generic protocol

Let us consider two sets of functions $(f_i)_{1\leq i\leq n}$ and $(g_i)_{1\leq i\leq n}$ such that the f_i do not lend themselves to an efficient verification with the GVC system whereas the g_i can be handled by the GVC system efficiently. For an input x, we denote by y the evaluation of x by the function $g_n \circ f_n \circ \ldots g_1 \circ f_1$. In our embedded proof protocol, each function f_i is handled by a sub-prover $\mathcal{P}_i$ while the g_i functions are handled by the prover $\mathcal{P}$. The sub-prover $\mathcal{P}_i$ is in charge of the efficient VC algorithm $\mathtt{EVC}_i$ and the prover $\mathcal{P}$ runs the GVC algorithm. The steps of the proof generation are depicted in Figure 4.2. Basically, each sub-prover $\mathcal{P}_i$ will evaluate the function f_i on a given input, produce

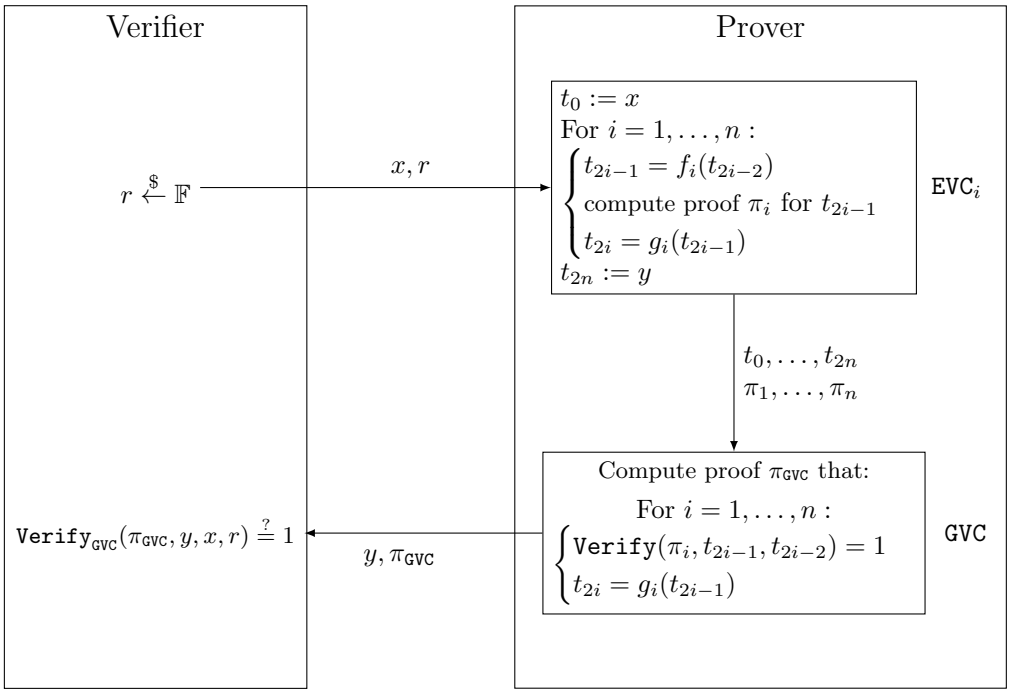


Figure 4.2 – Embedded proof protocol

a proof of correct evaluation using the $\mathtt{EVC}_i$ system and pass the output of f_i and the related proof π_i to $\mathcal{P}$, who will compute the next g_i evaluation and pass the result to the next sub-prover $\mathcal{P}_{i+1}$.

In the **Setup** phase, the verifier and the prover agree on an arithmetic circuit which describes the computation of the functions g_i along with the verification algorithms of the proof that the functions f_i were correctly computed. The pre-processing phase of the GVC system takes the resulting circuit and outputs the corresponding evaluation and verification keys.

In the **query** phase, the verifier sends the prover an input x for the computation along with a random value that will be an input for the efficient sub-provers $\mathcal{P}_i$.

In the **proving** phase, $\mathcal{P}_1$ first computes $t_1 = f(x)$ and produces a proof π_1 of the correctness of the computation, using the efficient proving algorithm $\mathtt{EVC}_1$. The prover $\mathcal{P}$ then computes the value $t_2 = g_1(t_1)$ and passes the value t_2 to $\mathcal{P}_2$, who computes $t_3 = f_2(t_2)$ along with the proof of correctness π_2, using the $\mathtt{EVC}_2$ proving system. The protocol proceeds until $y = t_{2n}$ is computed. Finally, $\mathcal{P}$ provides the inputs/outputs of the computations and the intermediate proofs π_i to the GVC system and, using the evaluation key computed in the setup phase, builds a proof $\pi_{\mathtt{GVC}}$ that for $i = 1, \ldots, n$:

1. the proof π_i computed with the $\mathtt{EVC}_i$ system is correct,

2. the computation $t_{2i} = g_i(t_{2i-1})$ is correct.

In the **verification** phase, the verifier checks that y was correctly computed using the GVC's verification algorithm, the couple $(y, \pi_{\mathtt{GVC}})$ received from the prover, and (x, r).

Recall that our goal is to gain efficiency compared with the proof generation of the whole computation inside the GVC system. Therefore, we need proof algorithms with

a verification algorithm that can be implemented efficiently as an arithmetic circuit and for which the running time of the verification algorithm is lower than the one of the computation. Since the Sum-Check protocol involves algebraic computations over a finite field, it can easily be implemented as an arithmetic circuit and fits into our scheme.

4.4.2 Protocol instance using Pinocchio and Sum-Check

In this section, we give a description of embedded proofs in the case where the functions f_i takes as input a matrix X and returns the product $W_i \times X$ and the functions g_i are functions that no efficient VC system but `GVC` can verify. We use the Sum-Check protocol to prove correctness of the matrix multiplications, as in [Tha13] and Pinocchio as the global proof mechanism. We assume that the matrices involved in the f_i functions do not have the same sizes so there will be several instances of the Sum-Check protocol. It thus makes sense to define different efficient proving algorithms $\texttt{EVC}_i$. Indeed, the Pinocchio system requires that the verification algorithms are expressed as arithmetic circuits in order to generate evaluation and verification keys for the system. As the parameters of the verification algorithms are different, the Sum-Check verification protocols are distinct as arithmetic circuits. For the sake of simplicity, the W_i matrices are assumed to be square matrices of size n_i. We assume that $n_i \geq n_{i+1}$ and we denote $d_i = \log n_i$. We denote by H the Ajtai hash function (see Section 4.3.3 for details).

The protocol is the following:

Setup:
- Verifier and Prover agree on an arithmetic circuit $\mathcal{C}$ description for the computation. $\mathcal{C}$ implements both the evaluations of the functions g_i and the verification algorithms of the Sum-Check protocols for the n matrix multiplications.
- $(EK_{\mathcal{C}}, VK_{\mathcal{C}}) \leftarrow \texttt{KeyGen}(1^{\lambda}, \mathcal{C})$

Query Verifier:

- generates a random challenge (r_L, r_R) such that: $(r_L, r_R) \in \mathbb{F}^{d_1} \times \mathbb{F}^{d_1}$
- sends the prover the tuple (X, r_L, r_R), where X is the input matrix of the computation.

Proof :

For $i = 1, \ldots, n$, **Sub-prover** $\mathcal{P}_i$ on input (T_{2i-2}, r_L, r_R):

- computes the product $T_{2i-1} = W_i \times T_{2i-2}$, (denoting $T_0 := X$)
- computes r_{L_i} and r_{R_i} by respectively selecting the d_i first component of r_L and r_R,
- computes the multilinear extension evaluation $\widetilde{T}_{2i-1}(r_{L_i}, r_{R_i})$
- computes, using serialized Sum-Check protocol, the proof π_i of the evaluation of the polynomial:

$$P_i(x) = \widetilde{W}_i(r_{L_i}, x) \cdot \widetilde{T}_{2i-2}(x, r_{R_i}) \tag{4.4}$$

  where $x = (x_1, \ldots, x_{d_i}) \in \mathbb{F}^{d_i}$.
- sends the tuple $(T_{2i-2}, T_{2i-1}, W_i, \pi_i, r_{L_i}, r_{R_i})$ to prover $\mathcal{P}$.

Prover $\mathcal{P}$ computes the value $T_{2i} = g_i(T_{2i-1})$ and sends (T_{2i}, r_L, r_R) to the next sub-prover $\mathcal{P}_{i+1}$

Prover $\mathcal{P}$, receiving from sub-provers the inputs $\{(T_{2i-2}, T_{2i-1}, W_i, \pi_i, r_{L_i}, r_{R_i})\}_{i=1,\dots,n}$:

- Computes $\widetilde{T}_{2i-1}(r_{L_i}, r_{R_i})$.
- Parses π_i as $(P_{i,1}, r_{i,1}, P_{i,2}, r_{i,2}, \dots, P_{i,d_1}, r_{i,d_i})$, where the proof contains the coefficient of the degree two polynomials $P_{i,j}$ that we denote by $(a_{i,j}, b_{i,j}, c_{i,j})$ if: $P_{i,j}(x) = a_{i,j}x^2 + b_{i,j}x + c_{i,j}$
- Verifies π_i:
  - Checks: $P_{i,1}(0) + P_{i,1}(1) \stackrel{?}{=} \widetilde{T}_{2i-1}(r_{L_i}, r_{R_i})$
  - Computes: $r_{i,1} = \left(\sum_j r_{L_i}[j]\right) \cdot \left(\sum_j r_{R_i}[j]\right)$
  - For $j = 2, \dots, d_i$,
    - Check: $P_{i,j}(0) + P_{i,j}(1) \stackrel{?}{=} P_{i,j-1}(r_{i,j-1})$
    - Computes: $r_{i,j}$ as the product of components of the Ajtai hash function output, i.e. $r_{i,j} = \prod_{k=1}^{3} H(a_{i,j}, b_{i,j}, c_{i,j}, r_{i,j})[k]$
  - From T_{2i-2} and W_i, computes the multilinear extensions: $\widetilde{W}_i(r_{L_i}, r_{i,1}, \dots, r_{i,d_1})$ and $\widetilde{T}_{2i-2}(r_{i,1}, \dots, r_{i,d_1}, r_{R_i})$
  - Checks that $P_{d_i}(r_{i,d_i})$ is the product of the multilinear extensions $\widetilde{W}_i(r_{L_i}, r_{i,1}, \dots, r_{i,d_i})$ and $\widetilde{T}_{2i-2}(r_{i,1}, \dots, r_{i,d_i}, r_{R_i})$.
- Aborts if one of the previous checks fails. Otherwise, accepts T_{2i-1} as the product of W_i and T_{2i-2}.
- Repeat the above instructions for all the inputs until the proof π_n has been verified.
- Using Pinocchio, computes the final proof $\pi_{\texttt{GVC}}$ that all the $\texttt{EVC}_i$ proofs π_i have been verified and all the T_{2i} values have been correctly computed from T_{2i-1}.
- Sends $(Y, \pi_{\texttt{GVC}})$ to the Verifier.

Verification Verifier:

- computes $\texttt{Verify}(X, r_R, r_L, Y, \pi_{\texttt{GVC}})$
- If `Verify` fails, rejects the value Y. Otherwise accepts the value Y as the result of:

$$Y = g_n(\dots(g_2(W_2(g_1(W_1 \cdot X))))\dots)$$

4.4.3 Prover's input privacy

Pinocchio is a QAP-based protocol (see Section 3.3) and Gennaro et al. proved in [GGPR13] that their QAP-based protocol is statistical zero-knowledge for input provided by the prover. The combination of the proof of knowledge and zero knowledge properties in the zk-SNARK proof enables the prover to provide some inputs for the computation to be proved with Pinocchio for which no information will leak. Thus in the complete example of figure 4.2, the verifier only knows t_0, i.e. x and t_{2n}, i.e. y which are the input output of the global operation. Intermediate inputs, such as $t_i, i = 1, \dots, 2n-1$, are hidden from the verifier even though they are taken into account during the verification of the intermediate proofs by the `GVC` prover. Therefore, thanks to the zk-SNARKs the intermediate results are verified but not disclosed to the verifier.

4.5 Embedded proofs for Neural Networks

4.5.1 Motivation

In order to show the relevance of the proposed embedded proof scheme, we apply the resulting scheme to Neural Networks (NN), which are machine learning techniques achieving state of the art performance in various classification tasks such as handwritten digit recognition, object or face recognition. As stated in Section 4.3.1, a NN can be viewed as a sequence of operations, the main ones being linear operations followed by so-called activation functions. The linear operations are modeled as matrix multiplications while the activation functions are non-linear functions. A common activation function choice is the ReLU function defined by: $x \mapsto \max(0, x)$. Due to the sequential nature of NNs, a simple solution to obtain a verifiable NN would consist of computing proofs for each part of the NN sequence. However, this solution would degrade the verifier's performance, increase the communication costs and force the prover to send all the intermediate results, revealing sensitive data such as the parameters of the prover's NN. On the other hand, even if it is feasible in principle to implement the NN inside a `GVC` system like Pinocchio, the size of the matrices involved in the linear operations would be an obstacle. The upper bound for the total number of multiplications Pinocchio can support as part of one application is estimated at 10^7 [WSR+15]. This threshold would be reached with a single multiplication between two 220×220 matrices. In contrast, our embedded proof protocol enables to reach much larger matrix sizes or, for a given matrix size, to perform faster verifications of matrix multiplications.

4.5.2 A use-case where input privacy is not required

Despite the fact that no efficient `GVC` built system can currently provide privacy for the verifier's input, verifiable computation of Neural Networks can still suit some real life scenarios. State of the art face recognition systems use NNs [SKP15, TYRW14] to extracts features from a face picture. The features are gathered in a vector, called a (biometric) template, which has the property that if another extraction is performed from the same individual, there will be a short Euclidean distance between the resulting vector and the reference template of the individual. Note that some other distance can also be considered. In contrast, extracting features based on another individual's face will yield a vector at a greater distance. Now, consider the case of e-passports: such documents contain a chip in which information on the passport holder is written, in particular the chip contains a digital picture of the holder. It could therefore be interesting to add into the chip a reference template obtained using Neural Networks. To increase the confidence in the resulting vector, a proof computed with a VC scheme could be added, which would then allow to check that the template has been correctly computed. In this use-case, the privacy of the verifier's input is not required since the information would already be stored in the passport chip.

4.5.3 A Verifiable Neural Network Architecture

We now describe how our proposed protocol can provide benefits in the verification of a neural network (NN): in the sequel, we compare the execution of a GVC protocol on a two layers NN with the execution of the embedded proof protocol on the same NN. Since NN involve several matrix multiplications, embedded proofs enable substantial

gains. See Section 4.7.2 for implementation report. We stress that we consider neural networks in the *classification* phase, which means we consider that all the values have been set during a *training* phase, using an appropriate set of labeled inputs.

The NN we verify starts with a fully connected layer combined with a ReLU activation layer. We then apply a max pooling layer to decrease the dimensions and finally apply another fully connected layer. The execution of the NN can be described as:

$$\boxed{\text{INPUT}} \rightarrow \boxed{\text{FC}} \rightarrow \boxed{\text{RELU}} \rightarrow \boxed{\text{MAX POOLING}} \rightarrow \boxed{\text{FC}}$$

The fully connected and the ReLU layers have been mentioned in Section 4.5. The fully connected layer takes as input a value and performs a dot product between this value and a parameter that can be learned, ofter called a weight. Gathering all the fully connected layer weights in a matrix, the operation performed on the whole inputs is a matrix multiplication. The ReLU layer takes as input a matrix and performs element-wise the operation $x \mapsto \max(0, x)$. We will denote $X \mapsto \text{ReLU}(X)$. The max pooling layer takes as input a matrix and return a matrix with smaller dimensions. This layer applies a max function on sub-matrices of the input matrix, which can be considered as sliding a window over the input matrix and taking the max of all the values belonging to the window. The size of the window and the number of inputs skipped between two mapping of the max function (called *stride*) are parameters of the layer but do not change during the training phase nor on the classification phase. Usually the stride value and the filter size are set to 2, therefore a 2×2 window slide over the input matrix, with no overlapping over the inputs. For a window size of f and a stride of s, we will denote MaxPool the function taking as input a $n \times n$ matrix and returning a $m \times m$ matrix such that: $m = (n - f)/s + 1$ and applying the max function to the $f \times f$ window.

Denoting by W_1 and W_2 the matrix holding the parameters of the fully connected layers, X the input matrix and Y the output of the NN computation, the whole computation can be described as:

$$Y = W_2 \cdot \text{MaxPool}(\text{ReLU}(W_1 \cdot X)) \tag{4.5}$$

The equation (4.5) can also be rewritten as a sequence of operations:

$$X \rightarrow T_1 = W_1 \cdot X \rightarrow T_2 = \text{ReLU}(T_1) \rightarrow T_3 = \text{MaxPool}(T_2) \rightarrow Y = W_2 \cdot T_3 \tag{4.6}$$

4.6 Cost evaluation

Regarding the embedded proofs protocol, we seek efficiency gains over the execution of the complete function evaluation using the GVC system. The `GVC` system used as part of the concrete instance in the cost study is Pinocchio and the main cost factor in this protocol is the number of multiplicative gates of the arithmetic circuit to verify. The main focus of the cost evaluation is thus the number of multiplications performed by the `GVC` prover $\mathcal{P}$.

Matrix multiplication cost.

For the example we developed in Section 4.4.2, we need to compare the cost of executing a matrix multiplication in the Pinocchio system with the cost of implementing the verification of the Sum-Check protocol inside Pinocchio. QAP encodes the constraints of all multiplication gates of the circuit to verify, hence the last operation performed

in the circuit cannot be an addition. In a matrix multiplication $C = A \cdot B$, each component $c_{i,j}$ of the product is the result of an operation $\sum_k a_{i,k} \cdot b_{k,j}$. Multiplications between components are first performed and the product component $c_{i,j}$ is the addition of the latter multiplications. The representation as a circuit has thus to add an extra multiplication gate, which is a multiplication by 1, to enable the verification of the last addition in the circuit constraints. Consequently, in the multiplication between $n \times n$ matrices, the computation of each of the n^2 component requires $n + 1$ multiplicative gates in the corresponding arithmetic circuit.

To sum up, a circuit implementing matrix multiplication between two $n \times n$ matrices requires $(n + 1) \times n^2 = n^3 + n^2$ multiplication gates.

Sum-check verification cost.

The verification algorithm of the Sum-Check protocol has three parts. The first one is the consistency checks for the received univariate polynomials, the second one is the computation of multilinear extensions to perform the last check of the protocol while the last one is the hash computation of the challenge in the serialized proof. In the sequel, we assume that the sub-prover $\mathcal{P}$ has received a proof $\pi_{\texttt{EVC}_1} = (P_1, r_1, P_2, r_2, \ldots, P_d, r_d)$ from sub-prover $\mathcal{P}_1$, where $d = \log n$. The same reasoning would apply for $\mathcal{P}_2$.
Consistency checks. Recall that the verifier has to check if:$P_1(0) + P_1(1)$ is equal to the claimed value and if $P_i(0) + P_i(1) = P_{i-1}(r_{i-1})$ for $i = 2, \ldots, d$. If sub-prover $\mathcal{P}_1$ sends the coefficient of the degree 2 polynomials P_i, which we will denote by a_i, b_i, c_i, then the check becomes:

$$\begin{aligned} a_i + b_i + 2c_i &= a_{i-1}r_{i-1}^2 + b_{i-1}r_{i-1} + c_{i-1} \\ &= (a_{i-1}r_{i-1} + b_{i-1})r_{i-1} + c_{i-1}. \end{aligned} \tag{4.7}$$

The cost of this check is 3 multiplicative gates for the equality testing (see [PHGR13]) and, using Horner algorithm, 2 multiplication gates for the $P_{i-1}(r_{i-1})$ evaluation, the computation of the left hand side being free. Adding the first check, which is only an equality check, we obtain a cost of $5 \cdot \log n + 3$ multiplicative gates for (4.7).
Multilinear extension computation. For the final check, sub-prover $\mathcal{P}$ has to compute the multilinear extension of the input matrices that we will denote by A, B, C for convenience. Suppose that A is a (n, n) matrix and denote $d = \log n$. A can be interpreted as a function $A : \{0, 1\}^{d \times d} \to \mathbb{F}$, associating to each index (i, j) value written in binary form the value $A(i, j)$. In the last step of the Sum-Check protocol, the verifier has to compute $\tilde{A}(r_L, r_1, \ldots, r_d)$, where $r_L = (r_{L_1}, \ldots, r_{L_d}) \in \mathbb{F}^d$ is the randomness sent to the prover. Cormode et al. [CTY11] describe an algorithm to compute this multilinear extension, using $O(n \times d)$ time and $O(d)$ space. Vu et al. [VSBW13] proposed an optimization of the previous algorithm, provided that the input data are not streaming data. At the expanse of $O(n)$ space, the multilinear extension computation decreases to $O(n)$ time. For convenience, we rewrite the randomness $(r_1, \ldots, r_{2d})$. The algorithm computes a table T which contains the values $\chi_{(w_1,\ldots,w_{2d})}(r_1, \ldots, r_{2d})$. Denoting $T^{(j)}$ the 2^j new values of the table that are computed at stage j, $T^{(j)}[(w_1, \ldots, w_j)]$ is computed using the fact that:

$$T^{(j)}[(w_1, \ldots, w_j)] = T^{(j-1)}[(w_1, \ldots, w_{j-1})] \cdot (w_j \cdot r_j + (1 - w_j) \cdot (1 - r_j)) \tag{4.8}$$

Therefore, each stage of the table computation requires 3 multiplications, for a total

of $\sum_{j=1}^{2d} 3 \cdot 2^j = 6n^2 - 6$ multiplications. As a comparison, using the Cormode et al. algorithm would require $6n^2 \log n$ multiplications.

Challenge computations in the proof serialization. The Sum-Check protocol, as performed for the matrix multiplication (Section 4.4.2), involves d rounds. The verifier need to compute a hash value to check the challenge of the next round is correct. Using Ajtai hash function with the parameters $m = 1524, n = 3$ (see Section 4.3.3), the challenge at round $i + 1$ is computed as the product of the $n = 3$ components of $H(a_i \parallel b_i \parallel c_i \parallel r_i)$. The computation of each component is a linear combination of the Ajtai matrix coefficients and since multiplication by constants are free in QAP-based VC schemes, only three constraints are required to verify the hash computation and three constraints to check the equality between the computed hash and the given challenge. Therefore all the challenges are verified using $6d$ gates.

Gathering all the sub-costs, the overall cost to verify the Sum-Check protocol for matrix multiplication is $18n^2 + 11 \log n - 15$ gates.

4.7 Implementation and Performance Evaluation

We ran two sets of experiments to compare the cost of execution between our embedded proof scheme and a baseline scheme using the `GVC` scheme. The first set focuses only on the cost of a matrix multiplication since these are a relevant representative of complex operations whereby the embedded proof scheme is likely to achieve major performance gains. The second set takes into account an entire application involving several operations including matrix multiplications, namely a small neural network architecture see Section 4.7.2 for an overall description.

4.7.1 Matrix multiplication benchmark

We implemented our embedded proof protocol on a 8-core machine running at 2.9 GHz with 16 GB of RAM. The GVC system is Groth state of the art zk-SNARK [Gro16] and is implemented using the `libsnark` library [lib] while the EVC system is our own implemention of Thaler's special purpose matrix multiplication verification protocol [Tha13] using the NTL library [Sho].

The proving time reported in table 4.1 measures the time to produce the proof using the EVC system and to verify the result inside the GVC system. For values of n higher than 256 the proof using the `GVC` is not feasible (denoted `unr.` in the table) whereas the embedded proof approach still achieves realistic performance.

Table 4.2 compares the key generation time using the embedded proof system with the one using the `GVC`. Table 4.3 states the sizes of the proving key (PK) and the verification key (VK) used in the previous scenarios.

Table 4.1 – Matrix multiplication proving time

n	16	32	64	128	256	512
Baseline (GVC only)	0.25 s	1.76 s	16.24 s	145.39 s	`unr.`	`unr.`
Embedded proofs	0.41 s	1.52 s	7.36 s	31.41 s	127.27 s	538.45 s

Table 4.2 – Matrix multiplication key generation time

n	16	32	64	128	256	512
Baseline (GVC only)	0.48 s	3.17 s	20.05 s	150.48 s	`unr.`	`unr.`
Embedded proofs	0.56 s	1.67 s	5.96 s	25.52 s	77.35 s	284.82 s

Table 4.3 – Matrix multiplication key generation size

n	16	32	64	128	256	512
Baseline (GVC only) PK	984 Ko	7.33 Mo	56.5 Mo	443 Mo	`unr.`	`unr.`
Embedded proofs PK	1.14 Mo	3.29 Mo	11.4 Mo	43.4 Mo	171 Mo	680 Mo
Baseline (GVC only) VK	31 Ko	123 Ko	490 Ko	1.96 Mo	`unr.`	`unr.`
Embedded proofs VK	32 Ko	124 Ko	492 Ko	1.96 Mo	7.84 Mo	31.36 Mo

4.7.2 Two-Layer Verifiable Neural Network Experimentations

We implemented the verification of an example of 2-layer neural network, which can be seen as one matrix multiplication followed by the application of two non-linear functions, namely a ReLU and a max pooling function as described in Section 4.5. For our experiments, the max pooling layers have filters of size 2×2 and no data overlap. Thus, setting for instance the first weight matrix to 64×64, the second weight matrix size is 32×32; we denote by NN-64-32 such a neural network. Table 4.4a reports experiments on a 2-layer neural network with a first 64×64 matrix product, followed by a ReLU and a max-pooling function, and ending with a second 32×32 matrix product. Experimental times for a NN-128-64 network (with the same architecture as above) are reported in table 4.4b.

	`KeyGen`	PK size	VK size	Prove	Verify
Baseline (GVC only)	59 s	148 MB	490 kB	25.48 s	0.011 s
Embedded proofs	44 s	123 MB	778 kB	16.80 s	0.016 s

(a) NN-64-32

	`KeyGen`	PK size	VK size	Prove	Verify
Baseline (GVC only)	261.9 s	701.5 MB	1.96 MB	149.5 s	0.046 s
Embedded proofs	162.7 s	490 MB	3.1 MB	66.96 s	0.067 s

(b) NN-128-64

Table 4.4 – Experiments on 2-layer networks

Experiments show a proving time twice better than using the baseline proving system. The overall gain is lower than for the matrix product benchmark because the other operations (ReLU and max pooling) are implemented the same way for the two systems. It should be noted that the goal of the implementation was to achieve a proof of concept for our scheme on a complete composition scenario involving several functions rather than putting in evidence the performance advantages of the scheme over the baseline, hence the particularly low size of the matrices used in the 2-layer NN and an advantage

as low as the one in table 4.4a and table 4.4b. The gap between the embedded proof scheme and the baseline using a realistic scenario with larger NN would definitely be much more significant due to the impact of larger matrices as shown in the matrix product benchmark.

4.8 Security Evaluation

Our embedded proof system has to satisfy the correctness and soundness requirements. Suppose that we have a GVC and n EVC systems to prove the correct computation of $y = g_n \circ f_n \circ \ldots \circ g_1 \circ f_1(x)$. We will denote by $\texttt{EVC}_i$, $i = 1, \ldots, n$ the EVC systems. We also keep notations defined in Section 4.4: the value t_i, $i = 0, \ldots, 2n$ represents intermediate computation results, t_{2i-1} being the output of the f_i function, t_{2i} being the output of the g_i function, $t_0 := x$ and $t_{2n} = y$. The above systems already satisfy the correctness and soundness requirements. Let denote by $\epsilon_{\texttt{GVC}}$ the soundness error of the GVC system and $\epsilon_{\texttt{EVC}_i}$ the soundness error of the $\texttt{EVC}_i$ system. Note that while the $\texttt{EVC}_i$ systems prove that $t_{2i-1} = f_i(t_{2i-2})$ have been correctly computed, the GVC system proves the correctness of $2n$ computations, namely that the *verification* of the $\texttt{EVC}_i$ proofs has passed and that the computations $t_{2i} = g_i(t_{2i-1})$ are correct. Furthermore, the GVC system proves the correct execution of the function F that takes as input the tuple $(x, y, r, (t_i)_{i=1,\ldots,2n}, (\pi_i)_{i=1,\ldots,n})$ and outputs 1 if for all $i = 1, \ldots, n$, $\texttt{Verify}_{\texttt{EVC}_i}(\pi_i, t_{2i-1}, t_{2i-2}) = 1$ and $t_{2i} = g_i(t_{2i-1})$. F outputs 0 otherwise. For convenience, we denote by comp_n the function $g_n \circ f_n \circ \ldots \circ g_1 \circ f_1$.

4.8.1 Correctness

Theorem 6. *If the* $\texttt{EVC}_i$ *and the* GVC *systems are correct then our embedded proof system is correct.*

Proof. Assume that the value $y = \text{comp}_n(x)$ has been correctly computed. It means that for $i = 1, \ldots, n$, the values $t_{2i-1} = f_i(t_{2i-2})$ and $t_{2i} = g_i(t_{2i-1})$ have been correctly computed. Since the GVC system is correct, it ensures that the function F will pass the GVC verification with probability 1, provided that its result is correct. Now, since the $\texttt{EVC}_i$ systems are correct, $\texttt{Verify}_{\texttt{EVC}_i}(t_{2i-1}, t_{2i-2}, \pi_i) = 1$ with probability 1.

Therefore, if $y = \text{comp}_n(x)$ has been correctly computed, then the function F will also be correctly computed and the verification of the embedded proof system will pass with probability 1. □

4.8.2 Soundness

Theorem 7. *If the* $\texttt{EVC}_i$ *and the* GVC *systems are sound with soundness error respectively equal to* $\epsilon_{\texttt{EVC}_i}$ *and* $\epsilon_{\texttt{GVC}}$*, then our embedded proof system is sound with soundness error at most* $\epsilon := \sum \epsilon_{\texttt{EVC}_i} + \epsilon_{\texttt{GVC}}$.

Proof. Assume that a p.p.t. adversary $\mathcal{A}_{emb}$ returns a cheating proof π for a result y' on input x, i.e. $y' \neq \text{comp}(x)$ and π is accepted by the verifier $\mathcal{V}_{emb}$ with probability higher than ϵ. We then construct an adversary $\mathcal{B}$ that breaks the soundness property of either the GVC or of one of the EVC systems. We build $\mathcal{B}$ as follow: $\mathcal{A}_{emb}$ interacts with the verifier $\mathcal{V}_{emb}$ of the embedded system until a cheating proof is accepted. $\mathcal{A}_{emb}$ then forwards the cheating tuple $(x, y, r, (t_i)_{i=1,\ldots,2n}, (\pi_i)_{i=1,\ldots,n})$ for which the proof π has been accepted. Since $y' \neq \text{comp}(x)$, there exists an index $i \in \{1, \ldots, n\}$ such that

either $t_{2i-1} \neq f_i(t_{2i-2})$ or $t_{2i} \neq g_i(t_{2i-1})$. $\mathcal{B}$ can thus submit a cheating proof to the $\texttt{GVC}$ system or to one of the $\texttt{EVC}_i$ system, depending on the value of the index i.

Case $t_{2i-1} \neq f_i(t_{2i-2})$

By definition of the proof π, this means that the proof π_i has been accepted by the verification algorithm of $\texttt{EVC}_i$ implemented inside the $\texttt{GVC}$ system. $\mathcal{A}_{emb}$ can then forward to the adversary $\mathcal{B}$ the tuple $(t_{2i-1}, t_{2i-2}, \pi_i)$. Now if $\mathcal{B}$ presents the tuple $(t_{2i-1}, t_{2i-2}, \pi_i)$ to the $\texttt{EVC}_i$ system, it succeeds with probability 1. Therefore, the probability that the verifier $\mathcal{V}_{emb}$ of the embedded proof system accepts is:

$$\begin{aligned} Pr[\mathcal{V}_{emb} \text{ accepts } \pi] &= Pr[\mathcal{V}_{\texttt{EVC}_i} \text{ accepts } \pi_i \mid \mathcal{V}_{emb} \text{ accepts } \pi] \\ &\times Pr[\mathcal{V}_{emb} \text{ accepts } \pi] \\ &= 1 \times \epsilon \\ &\geqslant \epsilon_{\texttt{EVC}_i} \end{aligned}$$

Thus $\mathcal{B}$ breaks the soundness property of $\texttt{EVC}_i$.

Case $t_{2i} \neq g_i(t_{2i-1})$

This means that the proof π computed by the $\texttt{GVC}$ system is accepted by $\mathcal{V}_{emb}$ even if $t_{2i} \neq g_i(t_{2i-1})$ has not been correctly computed. We proceed as in the previous case: $\mathcal{A}_{emb}$ forwards $\mathcal{B}$ the cheating tuple and the cheating proof π. The tuple and the proof break the soundness of the $\texttt{GVC}$ scheme because we thus have:

$$Pr[\mathcal{V}_{emb} \text{ accepts } \pi] = \epsilon \geq \epsilon_{\texttt{GVC}}$$

□

4.9 Conclusion

We designed an efficient verifiable computing scheme that builds on the notion of proof composition and leverages an efficient VC scheme, namely the Sum-Check protocol to improve the performance of a general purpose VC protocol, Pinocchio, in proving matrix multiplications. As an application, our scheme can prove the correctness of a neural network algorithm. We prove that our scheme is sound and give an efficiency evaluation. We stress that the composition technique described in the article can be extended to other efficient VC schemes and to an arbitrary number of sequential function evaluations, provided that they respect the requirements defined in Section 4.3.2.

Chapter 5

Verifiable Computation and Zero-knowledge Proofs

5.1 Motivation: short ZK proofs for NP computations

Some use-cases require that the prover hides parts of his inputs. One can think for example of the Naor-Yung protocol [NY90] that turns an encryption scheme secure against chosen plaintext attack into a scheme secure against ciphertext attack by encrypting the same message under two different keys and proving that the resulting ciphertexts encrypts the same message. Here the prover obviously cannot reveal the key used for encryption to the verifier. Zero-knowledge (ZK) seems to be an adequate property for such requirement, as long as the time to produce the proof does not degrade the efficiency of the protocol. The first practical zero-knowledge proofs were designed for algebraic statements, such as Schnorr's identification protocol [Sch89] that is a zero-knowledge proof of knowledge of a discrete logarithm and that leverages the algebraic structure of multiplicative groups in its design. From discrete logarithm several other practical ZK-proofs were designed, such as discrete logarithm equality [CP92] or proofs that a discrete logarithm lies in a given interval [Bou00]. These protocols also leverage algebraic structure and they have found practical applications due to their efficiency. While ZK proofs for algebraic statements have efficient instantiations, there have been for a long time few practical proposals of ZK proofs for generic statements. Yet, a zero-knowledge proof of knowledge of a hash function pre-image or a zero-knowledge proof of knowledge of an AES key are interesting and useful examples of such generic statements.

The lack of structure in such computations prevents from designing an ad hoc efficient ZK proof and generic transformations, as proposed e.g. in [GMW86], are too inefficient to be implementable in practice. Using secure two-party computation, Jawurek et al. [JKO13] designed the first efficient ZK proofs for non-algebraic computations. Despite an efficient proof computation, the drawback of their construction is the proof size and the necessity of communication between the prover and the verifier to establish the proof. Several state of the art VC schemes also achieve a zero-knowledge property that ensures the proof does not leak information about the prover's private inputs. QAP-based VC schemes like Pinocchio [PHGR13] or Buffet [WSR+15] notably require few extra work for the prover to get a zero-knowledge proof, turning the SNARK scheme into a zk-SNARK scheme. Some other state-of-the-art schemes such as Groth's scheme [Gro16] are zk-SNARKs by design. Compared to generic transformations, the relative

efficiency of zk-SNARKs enables to compute zero-knowledge argument of knowledge for computations that do not have a particular algebraic structure. And compared to Jawurek et al.'s proof system, the size of zk-SNARKs proofs is way shorter and require no communication. Moreover, when the prover can provide private inputs for a computation, the efficiency requirement described in chapter 1 makes no longer sense because the complete computation could not have been run by the sole verifier.

As a consequence of the above properties, zk-SNARKs schemes enable to design practical protocols based on the commit and prove paradigm, where a party first commits on a value and then proves that the committed value satisfies some property. Indeed, since zk-SNARK schemes can prove in zero-knowledge non-algebraic computations, the commitment can be instantiated with a hash function and a zero-knowledge proof that a private value hashes into the committed value can be computed. Such a proof avoids to reveal the committed value while providing guarantees that it indeed corresponds to the commit. In the sequel, we present two contributions we made, building primitives thanks to zk-SNARK schemes. Both contributions exploits the zero-knowledge property of zk-SNARKs to provide privacy for some inputs of the prover and build on the commit and prove paradigm.

The first zk-SNARK application we propose addresses the following problematic: is it possible to perform modifications on an authenticated document while keeping a notion of authenticity regarding the final document? Redactable signatures [JMSW02, SBZ01, BBD$^+$10] address this problem but they do not scale well regarding the signature size. Indeed, redactable signatures consider a message as a set of blocks that can be redacted separately and if the message has n blocks, then the signature is of size at least $O(n)$. In our proposal, we attach a proof generated by the mean of a zk-SNARK scheme to the modified document. By the succinctness property of zk-SNARKs, the proof has constant size, no matter the size of the document to modify. Hence, if the document we want to redact is a large image for which each pixel can potentially be redacted, the difference with redactable signatures is significant. In our protocol three parties are involved: a document issuer who is in charge of generating and authenticating the initial document, the client who receives the document from the document issuer and can perform modification on this document. The last party is the service provider who receives a redacted document and verifies its validity. In detail, the document issuer produces the original document, computes a hash value from the document and a random value and signs the hash. Then he passes the document, its signed hash and the random value used in the hash computation process to the client. The client can then redact some parts of the original document and keeps track of each modification performed as a set of blocks that have been modified. Once this operation is done, the client computes a proof that will be attached to the redacted document and that provides authenticity. Finally, the client passes the redacted document, the hash value and the set of locations of modification to the service provider. To verify the authenticity of the redacted document, the service provider checks that the hash value is authentic thanks to its signature; if the verification passes, the hash value and the redacted document are given as inputs to the proof verification algorithm. If the signature of the hash and the proof verification both pass, the service provider can be confident that the document is authentic. The proof is computed thanks to a zk-SNARK scheme and proves that:

- there exists a document that hashes into the given hash,
- the only differences between the original document and the redacted one lie in the

set of modification locations.

The public verifiability of existing zk-SNARKs schemes facilitate the possibility that multiple service providers adopt our verifiable redaction scheme. The zero-knowledge property guarantees that the proof will not reveal any information about the original document. Finally the zk-SNARK knowledge soundness property along with the unforgeability property of the signature scheme provide an unforgeability property to the scheme: an adversary not in possession of the original message cannot create a redacted document and a proof that will be accepted by the verifier.

Our second zk-SNARK application is a privacy-preserving biometric authentication protocol where, thanks to zk-SNARK, the user performs self-authentication and *proves* that the authentication process was correctly done. The protocol builds on the 'commit and prove' paradigm: the user commits to biometric templates and later proves that these committed values match. Thanks to the expressiveness of zk-SNARK schemes, it is possible to instantiate the commitment with hash functions and to recompute the hash value from the template. The proof of knowledge property of zk-SNARKs guarantees that if the proof is accepted, not only there exists matching templates but also the user is indeed in possession of them. Leveraging the zero-knowledge property of zk-SNARKs, the computation for which we prove correctness provides a privacy-preserving authentication: if an authentication is successful, the authentication server learns nothing except the fact that there exists two biometric templates – one that has been registered in front of a trusted authority and another one that has been freshly captured – that match and never gets any templates in clear. As a byproduct of self-authentication, the architecture of our authentication process is simplified compared to most of the privacy-preserving authentication schemes that split the traditional authentication server into a computational and an authentication server. The succinctness property of zk-SNARKs limits the communication cost with the authentication server: the user initializes the communication by sending his identification number, the server then only needs to reply by sending a nonce to avoid replay attacks and finally receives a proof of correct authentication and a commitment to a fresh template.

5.2 Verifiable Document Redacting

5.2.1 Problem Statement

People are frequently asked for information such as their place of residence, a source of income or a proof of employment in order to get e.g. a traveling visa or an identity card. They can provide a document, called a *breeder document*, which will be accepted as a proof as long as the document provider is trusted by the service which needs the paper. Nevertheless, these documents might contain private information that the owner does not want to share with the service provider asking for a justification. The problem addressed in this paper is to determine whether it is possible to keep sensitive information private on a document while giving a third party assurance that the redacted document was built from an authentic one.
To illustrate the relevance of the latter problem, we give below some examples where a document contains private information useless for the required justification:

- giving a pay stub to justify employment indeed gives the name and address of the employer but also reveals a sensitive and useless information for this goal, namely

the salary amount,

- someone can prove he has earnings by providing the balance of his bank statement (in order to get a visa for instance) but the detail of all the transactions written in the statement does not concern the entity needing a revenue justification,
- in some countries, for the issuance of documents like driver license or identity card, an individual has to prove his place of residence with a bill (e.g. an electricity bill) where his name is written. However, the bill can also mention the name of the partner, which has no connection with the original request.

Even if removing the sensitive information from the document looks as a natural and efficient solution to our problematic, service providers fear fraudulent document forgery and often ask to bring the original document.

In the sequel, we argue that documents digitization opens the possibility to use cryptographic techniques such as signature to guarantee integrity and authenticity of the document issued by the trusted provider. Still, a problem remains: if the user makes redaction on a signed document, the signature cannot be verified with the new modified document. The client could ask the document issuer to edit a new redacted and signed version of the original document but this reveals which information is sensitive and thus is a privacy loss.

5.2.2 Related work: Redactable Signatures and Photoproof

In France, the most recent proposition to secure breeder document is called 2D-DOC [2dd]. It is a protocol to secure physical breeder document such as electricity bill, bank statement or phone bill. The most relevant information of the document are gathered and form a blob that is digitally signed. The blob and its signature are represented as a 2D bar-code and printed on the document.This guarantees the authenticity and integrity of the document. However, if the document is redacted the signature is no longer valid with the information left. Moreover, since the 2D bar-code contains the most relevant information of the document, private data appear on the bar-code and redacting the bar-code destroys the authenticity proof of the document.

Photoproof [NT16] is a recent protocol enabling the authentication of images that have been modified from an original one as long as the transformations belong to a well defined set. It builds on the notion of *proof carrying data* (PCD) [CT10] which are data along with a proof of some property satisfied by the data. PCD enable a data to be sequentially modified, the proof containing a proof of the current property and also a proof that all the previous data modifications have satisfied the required properties. PCD can be instantiated but the computational overhead for the prover is consequent: for example in Photoproof [NT16], limiting the set of transformation to cropping, rotating, transposing, bit flipping and modifying the brightness of the image, the authors report 300 seconds to build a proof for a 128×128 (pixels) image. The size of the public key used to build the proof is 2 GB; in contrast the verification is less than half a second long. So, even if the requirement of integrity and of confidentiality are satisfied, there is a need to simplify the above scheme in order to reach some efficiency and to be able to deal with larger images. Indeed, an A4 format bill scanned at 100 dpi produces a 1169×827 image. Our scheme also enables image authentication, but since we only allow redaction, we obtain much better proving time. Our scheme can therefore more easily scale on image size. See Sect. 5.2.3 for implementation results.

Redactable signatures are strongly related to our proposal. A redactable signature allows a party to remove parts of a signed document and to update the signature without possession of the signer's secret key. Moreover, the validation of the updated signature is still possible with the signer's public key. Redactable signatures have been independantly introduced by [SBZ01] and [JMSW02]; there has been a large body of work since, e.g. [BBD+10, SR10, DPSS15]. Our proposal shares some security goals with redactable signatures such as privacy of the redacted content and unforgeability of the signature. A notable difference is that everyone can redact a document in redactable signatures schemes while in our protocol only the owner of the document can perform redaction. Indeed some private inputs of the proof computed by the redactor cannot be supplied unless being in possession of both the original document and some value used to compute the hash. Our protocol enables redacting an image, a use case for which the existing redactable schemes would be impractical due to the length of the obtained signature or the time to generate the signature. Indeed in redactable signature schemes the length of the signature depends on the number of message blocks n and has at best a length of $\mathcal{O}(n)$. In the redaction of an image each pixel can be potentially redacted and therefore a block for an image to redact is a pixel. In contrast, our redacted signature has constant size. We finally note that our scheme cannot satisfy the transparency property as defined in [BBD+10], which states that it should be unfeasible to decide whether a signature directly comes from the signer or has been generated after some redaction. Indeed, we give places where redaction happened to the verifier and thus transparency cannot be reached. This fits however to our use case since the redacted document is given to the verifier and redacted places are thus visible to the verifier.

5.2.3 Our Solution

We propose a protocol to issue a redacted document from an original and authenticated document. Our protocol involves three parties: the issuer of the original document, the client that wants to redact the document and the document user who makes a request to the client. The protocol gives strong guarantees that nothing has been modified from the original document except the redacted parts. Moreover it links the redacted document to the original one while keeping the original one private. It should also be noticed that the issuer of the document has minimal work to do: after generating a document, he only has to compute a hash value and a signature. No further work is needed during the redaction or the verification of the document. The main tool for building this protocol is verifiable computation, see Chapter 3 for background and details.

Basically, our protocol is the following: the issuer first generates a document, signs a hash computed from the original document and sends the document, the hash and its signature to the client. The client then redacts some parts of the document and computes a non-interactive zero-knowledge proof, proving that only the redacted parts have been modified from the original document. The signature of the redacted document consists of the original signature and the proof. This signature has constant size and thus does not depend on the proportion of the document that has been redacted. The zero-knowledge property of the proof ensures that no information about the original document is contained in the proof. The client can then send the redacted document and its signature to the document user, along with some other elements needed to verify the proof. If the proof is correct, the document user can accept the redacted document with confidence. We stress that the document is publicly verifiable: the scheme produces

verification keys and anyone with access to these keys can verify the validity of the proof. Moreover since the proof has a constant short size, the verification is quick.

Let the *document issuer* (DI), the *client* (CL) and the *service provider* (SP) be the three parties involved in the scheme. The document issuer first generates a document $\mathcal{D}$, computes a hash value C from $\mathcal{D}$ and a random value r. DI then signs the hash value to authenticate it and sends the client $\mathcal{D}, C, r$ and the signature of C. To give the possibility to redact the document to the client while keeping a link with the original document, we use a verifiable computation scheme to produce a proof of the statement below. In the statement, MOD is a set describing all the redacted places of the document $\mathcal{D}$. In our motivating example, MOD would be the coordinates of all the pixels of the image that are turned black.

There exists a document $\mathcal{D}$ and a set of coordinates MOD such that the redacted document $\mathcal{D}_{red}$ only differs from $\mathcal{D}$ in places defined by the set MOD.

In the proof, the original document $\mathcal{D}$ stays private using a property of verifiable computing schemes: the prover can supply a private input in the computation and build a zero-knowledge proof of the computation. The verifier thus cannot infer information about the prover's input by examining the proof. To ensure that the proof has been built with the original document, a hash computed from the original document is added to the computation. Since this hash will be sent to the verifier of the redacted document it cannot be only the hash of the document, otherwise this would give an oracle for the verifier to test the redacted parts of the document. This is why the random value r is computed by the document issuer and concatenated to the document before the hash computation. Using the same notations, the statement to be proved now becomes:

There exists a document $\mathcal{D}$ and a value r such that the hash of $\mathcal{D} \parallel r$ equals C and such that $\mathcal{D}_{red}$ only differs from $\mathcal{D}$ in places defined by the set MOD.

Denoting by π the proof, the client thus passes $\pi, \mathcal{D}_{red}, MOD, C$ and its signature σ to SP. The service provider first verifies that the signature σ of C is correct to be sure that the hash of the original document is authentic. He then uses $C, \mathcal{D}_{red}$ and MOD to verify the proof π. We stress that π ties the hash value computed and authenticated by the document issuer to the original document because it proves (in zero-knowledge) that this document, concatenated with the value r hashes into C. Thus, the correct verification of the signature of C and of the proof π guarantees that the original document is authentic. In the next section, we give a more formal description of the scheme.

The verifiable document redacting protocol

In this section we define the syntax and the security of our scheme. As it was mentioned in the introduction, the security goals of our scheme are close to the redactable signatures goals [SBZ01].

Protocol syntax. Let $(\texttt{Gen}, \texttt{Sign}, \texttt{Ver})$ be a signature scheme [BS], H be a hash function and let the triple of algorithms $(\texttt{Setup}, \texttt{Prove}, \texttt{Verify})$ be a zk-SNARK [BCG$^+$13]. See Sect. 3.3.3 for details on the zk-SNARK algorithms.
The protocol participants are the Document Issuer (DI), the Client (CL) and the Service Provider (SP). Let $M = (m_1, \ldots, m_n)$ be a message composed of n sub-messages. We use a special symbol # to denote the redaction of a sub-message. When a message M

is redacted, the resulting message is denoted $M_{red} = (m_1^{red}, \ldots, m_n^{red})$. Our verifiable document redacting (**VDR**) scheme is a tuple of four polynomial time algorithms:

KeyGen$(1^\lambda, \mathcal{F})$: this probabilistic algorithm takes a security parameter λ and runs the `Gen` algorithm to output a secret/public signing key pair (SK, PK). It then takes λ and an arithmetic circuit over a finite field $\mathbb{F}_p$, runs the `Setup` algorithm and outputs a pair of public proving and verification keys $(EK_{\mathcal{F}}, VK_{\mathcal{F}})$ for the circuit $\mathcal{F}$.

Authent(M, SK) : this probabilistic algorithm, run by DI, takes a document M, a secret signing key SK and computes:

- $r \xleftarrow{\$} \{0,1\}^{128}$
- $C \leftarrow H(M \parallel r)$
- $\sigma \leftarrow \texttt{Sign}(C, SK)$

Output: (C, r, σ)

Redact$(M, C, r, \sigma, EK_{\mathcal{F}})$: this probabilistic algorithm, run by CL, takes a document M, the output (C, r, σ) computed by `Authent` and the evaluation key $EK_{\mathcal{F}}$ and computes:

- $d \leftarrow \texttt{Ver}(C, \sigma, PK)$
- If $d = 0$, then abort.
- Else:
  - define the set MOD of the redacted sub-messages, MOD is a subset of $\{1, \ldots, n\}$,
  - define M_{red} such that: $\forall i \in MOD, M_{red}(i) = \#$
  - $\pi \leftarrow \texttt{Prove}(([M, r], C, M_{red}, MOD), EK_{\mathcal{F}})$, where the value between brackets, namely the original document and the randomness used to compute C, are privately supplied by the prover and the circuit $\mathcal{F}$ used in `Prove` is built to verify the following statement:

$$\begin{cases} \exists M, r \text{ such that} H(M \parallel r) = C \\ \forall j \in MOD, m_j = \# \\ \forall j \notin MOD, m_j = m_j^{red} \end{cases}$$

Output: $(M_{red}, MOD, C, \sigma, \pi)$ – the signature of M_{red} is the pair (σ, π).

DocVerif$(M_{red}, MOD, \sigma, \pi, VK_{\mathcal{F}}, PK)$: this deterministic algorithm, run by SP, takes a redacted document M_{red}, a set of redacted sub-messages index MOD, a signature σ, a proof π and the signing public key and the verification key. It outputs a bit d such that:

- $d \leftarrow \texttt{Ver}(C, \sigma, PK)$
- $d \leftarrow d \times \texttt{Verify}(\pi, (M_{red}, C, MOD), VK_{\mathcal{F}})$

Protocol security. We now define the security goals of our scheme, adapting security notions defined in [BBD+10]. Our first goal is to reach privacy of the redacted document, informally meaning that no PPT adversary only in possession of the redacted message and its proof can recover information about the redacted parts of the message. Our second goal is unforgeability of the proof: a PPT adversary not being in possession of the original message cannot create a redacted document and a proof that will be accepted by the verifier. We formalize these goals below.

Privacy : a **VDR** scheme $(\texttt{KeyGen}, \texttt{Authent}, \texttt{Redact}, \texttt{DocVerif})$ is private if for all PPT adversaries $\mathcal{A}$, the probability that the experiment Leak evaluates to 1 is negligibly close to $\frac{1}{2}$.

The Leak experiment:

- $b \leftarrow \{0,1\}$
- $(M^0, M^1, i) \leftarrow \mathcal{A}$
  with (M^0, M^1, i) such that $\forall j \neq i,\ M_j^0 = M_j^1$ and $M_i^0 \neq M_i^1$
- $(M_{red}^b, C_b, \sigma_b, \pi_b) \leftarrow \mathcal{O}^{\texttt{Auth/Redact}}$
- $b^\star \leftarrow \mathcal{A}(PK, EK_{\mathcal{F}}, VK_{\mathcal{F}}, M_{red}^b, C_b, \sigma_b, \pi_b)$
- Return 1 if $b^\star = b$

The adversary's advantage is defined as: $\mathsf{Adv}_{\mathsf{Leak}}^{\mathcal{A}} = \left|\Pr[\mathsf{LeakExp} = 1] - \frac{1}{2}\right|$
A **VDR** scheme is private if $\mathsf{Adv}_{\mathsf{Leak}}^{\mathcal{A}}$ is negligible for all PPT adversaries.

Unforgeability : a **VDR** scheme $(\texttt{KeyGen}, \texttt{Authent}, \texttt{Redact}, \texttt{DocVerif})$ is unforgeable if for all PPT adversaries $\mathcal{A}$, the probability that the experiment Forge evaluates to 1 is negligible.

The $\mathsf{Forge}(\lambda)$ experiment:

- $(SK, PK, EK_{\mathcal{F}}, VK_{\mathcal{F}}) \leftarrow \texttt{KeyGen}(\lambda)$
- For $i = 1, \ldots, q$: $(M_{red}^i, MOD^i, \sigma^i, \pi^i) \leftarrow \mathcal{O}^{\texttt{Auth/Redact}}$
- $(M_{red}, MOD, \sigma, \pi) \leftarrow \mathcal{A}$
- Return 1 if:
  - $\texttt{DocVerif}(M_{red}, MOD, \sigma, \pi, VK_{\mathcal{F}}, PK) = 1$ and
  - $(M_{red}, MOD, \sigma, \pi) \neq (M_{red}^i, MOD^i, \sigma^i, \pi^i), \forall i \in \{1, \ldots, q\}$.

We define the advantage of the adversary as: $\mathsf{Adv}_{\mathsf{Forge}}^{\mathcal{A}} = |\Pr[\mathsf{ForgeExp} = 1]|$ The **VDR** scheme is unforgeable if $\mathsf{Adv}_{\mathsf{Forge}}^{\mathcal{A}}$ is negligible for all PPT adversaries.

Definition 15. *A **VDR** scheme is secure if it is private and unforgeable as defined above.*

Theorem 8. *If the signature scheme is existentially unforgeable under chosen message attack (EUF-CMA), the verifiable computing scheme is secure and the hash function is such that $H(., r)$ is a secure PRF then the **VDR** scheme is secure.*

Proof. For the sake of exposition, the proof details are given in Section 5.2.4. □

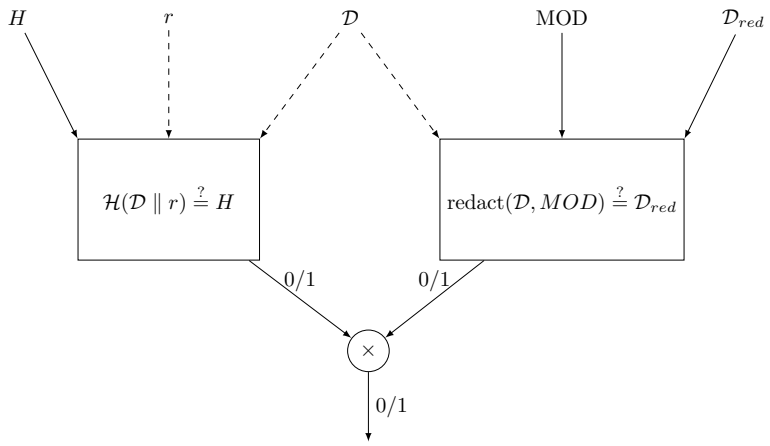


Figure 5.1 – Arithmetic circuit computing the proof in `Redact`. A dashed arrow means that the input is private (and supplied by the prover).

An instantiation of the VDR scheme

We now introduce a possible instantiation of the **VDR** scheme, keeping in mind that we seek efficiency for the prover. We consider that documents are represented as gray-scale images, modeled as matrices of $n \times n$ pixels. Pixels values vary between 0 (black) and 255 (white). Redacting a part of the document thus means that pixels of the redacted area are turned black, so the symbol # defined in Sect. 5.2.3 is the pixel value 0. The set MOD of redacted parts of the image is therefore a set of coordinates, which locates the redacted pixels positions.

We consider implemented verifiable computation schemes to instantiate our scheme, more specifically the scheme base on Parno et al. protocol [PHGR13, BCG$^+$13]. The verification is efficient and the schemes based on QAPs (see Sect. 3.3.2 for details) have a short, constant-length proof that is quick to verify. The difficulty is the prover's computational overhead, which is linked to the number of multiplication gates in the arithmetic circuit representing the function to verify. More precisely, the prover's work has complexity $O(N \log^2 N)$, where N is the circuit size [PHGR13]. Therefore an efficient arithmetic circuit has first to be designed to limit the number of multiplicative gates.

The arithmetic circuit design.

To build the proof used in the `Redact` algorithm of the **VDR** scheme (Sect. 5.2.3), an arithmetic circuit representing the computation to verify has to be designed in order to apply the Parno et al. protocol [PHGR13] (the description of this protocol can be found in Sect. 3.3.3). A high level view of this circuit is described in Fig. 5.1. It contains two sub-circuits verifying respectively the value of the hash passed by the document issuer to the client and the comparison between the original and the redacted documents. The operations involved in the sub-circuits are crucial for the prover efficiency. The circuit has to be carefully designed to be able to redact several document on different places and to amortize the key generation cost over several proof computation. Moreover, if the circuit which verify the correct redaction in the `Redact` algorithm is changed for some document, the evaluation and verification keys will change and have thus to be

exchanged with the service provider. We designed a circuit able to prove the correct redaction for every document modeled as an image of size $n \times m$.

The verification of the hash signature used by the document issuer is not part of the proof for efficiency reasons. Backes et al. [BBFR15] present a verifiable computing scheme suited for working with authenticated data but, even if the performance are better than verifiying signature with the Pinocchio scheme [PHGR13], the verification of the signature is way more efficient if it is done outside the proof. Besides, since the proof requires the hash of the document there is an explicit link between the redacted document and the original one. The addition of the hash to the proof only slightly increases the length of the proof. The verifier thus first verify the signature of the hash value to test whether it indeed correspond to a value generated by the document issuer. If the verification passes, the verifier can use this hash value as input for the proof verification.

Document Redaction Since the document to redact is modeled as a matrix, the proof described in Sect. 5.2.3 can be represented as an arithmetic circuit using a boolean matrix for the MOD set. The function in the verifiable computing scheme for which we compute a proof takes as input a redacted document $\mathcal{D}_{red}$, a hash value H and a set MOD. We denote by $d_{i,j}$ (resp. $d_{i,j}^{red}$) the pixel in position (i,j) of $\mathcal{D}$ (resp. $\mathcal{D}_{red}$). The prover supplies as private input the document $\mathcal{D}$ and the value r, the function f returns the value $d \in \{0,1\}$ which is the product of the following boolean tests:

$$\begin{cases} \exists r, \mathcal{D} \text{ such that:} & C \stackrel{?}{=} H(\mathcal{D} \parallel r) \\ \forall (i,j) \in MOD: & d_{i,j}^{red} \stackrel{?}{=} 0 \\ \forall (i,j) \notin MOD: & d_{i,j} - d_{i,j}^{red} \stackrel{?}{=} 0 \end{cases}$$

Using a boolean matrix M as a mask, we can rewrite the two last set of tests in a more uniform way. We define the matrix $M = (m_{i,j})$ as: $m_{i,j} = 0$ if pixel (i,j) is redacted and $m_{i,j} = 1$ otherwise. The tests can thus be rewritten as: $\forall (i,j) \in \{1,\ldots,n\}^2, d_{i,j} \times m_{i,j} \stackrel{?}{=} 0$. This leads to a small arithmetic sub-circuit to check if the redacted document has not been modified in other places that the given ones.

Hash function. The proof computed by `Redact` contains the verification of the hash value computed from the original document so we need to choose a hash function efficiently verifiable i.e. a function which can be represented as an arithmetic circuit with few gates. Hash function building on the subset sum problem are well suited for arithmetic circuits [BFR+13b, BCTV14a]. We also used another finite field in our experiments with a lower security level of 80 bit for the associated elliptic curve. Following the method of [KZM+15], we obtained the following parameters:

$$n = 2, m = 724, q = p \approx 2^{181}.$$

Few gates are needed to implement an arithmetic circuit for this hash function: to hash m bits, $n \times m$ multiplicative gates are needed. With the parameters selected in [KZM+15], this means that 4572 gates are needed to hash 1524 bits. As a comparison, Ben-Sasson et al. designed a hand-optimized arithmetic circuit to verify the compression function of SHA-256 [BCG+14]. Their arithmetic circuit can therefore hash 512 bits and has about 27000 gates.

Table 5.1 – Benchmark of verifiable computation in the **VDR** scheme (128 × 128 images, machine 1)

Security	Hash fct	Constraints	EK size	VK size	`KeyGen`	`Redact.Prove`	`DocVerif`
128 bit	Ajtai	19435	7.1 MB	1.3 MB	5.6 s	3.4 s	0.07 s
128 bit	SHA256	43920	13.7 MB	1.3 MB	9.2 s	4.7 s	0.07 s
80 bit	Ajtai	17834	5.4 MB	1.0 MB	5.5 s	2.4 s	0.07 s
80 bit	SHA256	43920	10.8 MB	1.0 MB	9.7 s	3.5 s	0.07 s

Experimental results

We implemented our protocol and benchmarked the verifiable computing part of the scheme since time consumption of the other parts is negligible compared to this one. Verifiable computation is implemented using the `libsnark` library [lib]. The tests were run on a two different machines. The first one, denoted by machine 1 in the tables, is running at 3.6 GHz with 4 GB of RAM, with no parallelisation. The second one, denoted machine 2, is more powerful: it has 8 cores running at 2.9 GHz with 16 GB of RAM and uses parallelisation. We first implemented our scheme for images of size 128 × 128 and chose elliptic curves at a 128 bit and 80 bit security level [BCTV14b]. The size of the proof is constant and short (less than 300 bytes) and thus the verification is fast. Table 5.1 summarizes the implementation results with machine 1. The column Constraints reports the number of constraints needed to check the satisfiability of the circuit implementing the proof redaction. For each security level of the proof, we implemented our scheme with the SHA256 hash and the Ajtai hash functions for comparison.

Table 5.2 reports implementation of the proving scheme using Ajtai hash function and a soundness security of 80 bit with variation on the image size. Table 5.3 reports the same implementation running on machine 2, with parallelisation. Note that even if the proof has constant size, the verification time increases with the image size. This is due to the time to parse the input redacted image to compute some elements to verify the proof. We continued our experiments until we reached approximately the size of an A4 document scanned at 100 dpi: we tested a 1200 × 800 image while the true size of an A4 document scanned at 100 dpi would be 1169 × 827.

The prover has most of the computational work to do with the `Redact` algorithm. However, this does not affect the practicality of the **VDR** scheme in the case of image redaction. Indeed, the proof is non-interactive and the client can prepare its redacted document, compute the related proof and submit both later to a service provider. On the service provider side, the verification is fast and does not require to share any secret with the client. The time to verify the signature of the hash value C has to be added to the verification time given in Table 5.1. Using a simple benchmark on OpenSSL, the time reported for signature verification is less than 1 ms for RSA signatures and ECDSA signatures with the same computer. We conclude that the **VDR** scheme is compatible with a practical use.

Table 5.2 – Scaling experiment of the proving part in the **VDR** scheme (machine 1 + no parallelisation)

Image size	Constraints	EK size	VK size	`KeyGen`	`Redact.Prove`	`DocVerif`
128×128	17834	5.4 MB	1.0 MB	5.6 s	2.4 s	0.07 s
400×400	161450	47.9 MB	9.9 MB	38.8 s	20.7 s	0.51 s
500×500	251450	74.0 MB	15.5 MB	58.2 s	32.8 s	0.89 s
600×600	361450	106.0 MB	22.3 MB	81.1 s	50.8 s	1.3 s
1200×800	961450	286.4 MB	59.5 MB	201.3 s	124.5 s	3.3 s

Table 5.3 – Scaling experiment of the proving part in the **VDR** scheme (machine 2 + parallelisation)

Image size	Constraints	EK size	VK size	`KeyGen`	`Redact.Prove`	`DocVerif`
128×128	17834	5.4 MB	1.0 MB	1.9 s	0.8 s	0.07 s
400×400	161450	47.9 MB	9.9 MB	12.4 s	5.9 s	0.5 s
500×500	251450	74.0 MB	15.5 MB	19.7 s	9.9 s	0.82 s
600×600	361450	106.0 MB	22.3 MB	26.2 s	14.5 s	1.17 s
1200×800	961450	286.4 MB	59.5 MB	66.8 s	39.7 s	3.3 s

5.2.4 Security Proofs

We prove Theorem 8 in this section. We will prove that our **VDR** scheme is private (Lemma 2) and unforgeable (Lemma 3), which will imply Theorem 8.

Lemma 2. *If the VC scheme provides (statistical) zero-knowledge proofs and the hash function H is such that $H_r := H(., r)$ is a secure PRF then the **VDR** scheme is private.*

Proof. We will bound the advantage of a PPT adversary attacking the privacy of the scheme using a sequence of games. More precisely we will show that $\mathtt{Adv}^{\mathcal{A}}_{\text{Leak}}$ is negligible.

Game 0 This is the original `Leak` game.

Game 1 Same as Game 0 but here the oracle $\mathcal{O}^{\mathtt{Auth/Redact}}$ picks a random value h, signs it and returns the couple h, σ, instead of C_b, σ. Let S_1 be the event that $b^\star = b$ in Game 1. Since $H(., r)$ is assumed to be a secure PRF, we have that: $Pr[S_0] - Pr[S_1] \leqslant \epsilon_{PRF}$, where ϵ_{PRF} is the PRF advantage.

Game 2 Same as Game 1, but the part of the oracle $\mathcal{O}^{\mathtt{Auth/Redact}}$ computing the proof is replaced by the simulator. Let S_2 be the event that $b^\star = b$ in Game 2. We have that $Pr[S_2] = Pr[S_1]$

Game 3 Same as Game 2, but the simulator of the oracle $\mathcal{O}^{\mathtt{Auth/Redact}}$ outputs its proof π without having knowledge of the messages $M_c, c \in \{0, 1\}$. Let S_3 be the event that $b^\star = b$ in Game 3. Since the VC scheme is assumed to be zero-knowledge, we have that there exists a negligible function ϵ_{SZK} such that: $Pr[S_3] - Pr[S_2] \leqslant \epsilon_{SZK}$. Since the signature is now only composed of random elements, we have $Pr[S_3] = \frac{1}{2}$.

Gathering the results of all the games, we finally conclude that:

$$|Pr[S_0] - \frac{1}{2}| \leqslant \epsilon_{PRF} + \epsilon_{SZK} \tag{5.1}$$

Therefore the **VDR** scheme is secure. □

□

Lemma 3. *If the VC scheme is sound and the signature scheme is EUF-CMA, then the **VDR** scheme is unforgeable.*

sketch. We show if there exists an efficient adversary succeeding in the `Forge` experiment, denoted by $\mathcal{A}_{\texttt{Forge}}$, we can build an efficient adversary $\mathcal{A}_{\texttt{EUF-CMA}}$ breaking the EUF-CMA property of the signature or an efficient adversary $\mathcal{A}_{\texttt{VC}}$ breaking the soundness of the verifiable computing scheme. These adversaries are built by forwarding the queries made by $\mathcal{A}_{\texttt{Forge}}$. At the end, $\mathcal{A}_{\texttt{Forge}}$ outputs a redacted forged document, which is not part of the queries made before. This redacted forged document is also a forgery for the signature scheme or for the verifiable computation scheme.

□

5.3 Privacy-preserving Biometric Commitments

5.3.1 Introduction

Biometrics and Authentication

Biometrics are a convenient way for a user to authenticate because, unlike password-based authentication, they do not require to remember any secret. A biometric authentication system usually proceeds with the steps described below.
During the *enrollment* phase, a user presents a biometric trait in front of a sensor. Then a signal processing algorithm (also named feature extraction algorithm) extracts useful information from the biometric reading, this information is gathered in a vector called a *biometric template* and the extracted template is stored as the reference template for future comparison.
During the *authentication* phase, a sensor (possibly different from the one that performed the enrollment) extracts information from the biometric trait presented. The feature extraction algorithm that was used during the enrollment then produces a fresh template, which is compared with the reference template. Depending on the result of the comparison, the system decides if the fresh biometric trait comes from the same individual.

It is important to note that repeating the acquisition of the same biometric trait with the same feature extraction algorithm may end with different templates. These variations can be explained by the way the trait is presented in front of the sensor and environment conditions (e.g. moisture or light conditions). As a consequence, two templates cannot be compared with a simple equality check. Therefore, a score has to be computed between the two templates to decide if they come from the same individual. This score is often the result of a distance computation and the definition of the distance is dependent of the feature extraction algorithm that processed the biometric trait.

To assess the performance of a biometric authentication system, the capacity to discriminate between individuals is a major factor. This capacity depends mainly on

the choice of the biometric modality and on the quality of the related feature extraction algorithm. A distance threshold τ is set and the system considers that two templates for which the distance is inferior to that threshold belong to the same individual, while the ones for which the distance is superior to τ are considered not to belong to the same individual. This raises two kind of errors regarding the system:

- Two templates coming from different individuals for which the distance is inferior to τ generate a *false accept*.
- Two templates coming from the same individual for which the distance is superior to τ generate a *false reject*.

The overall performance of a biometric system is thus measured through the *false acceptance rate* (FAR), which is the probability that a wrong template is accepted as genuine and the *false rejection rate* (FRR), which is the probability that a genuine template is rejected. Note that the two rates are linked: if a system threshold is modified to decrease the FAR, the FRR increases at the same time. Examples of biometric modalities that have been extensively studied and that perform well are iris, fingerprint and face. The associated distances are Hamming distance for iris recognition, Euclidean distance for facial recognition and for some fingerprint recognition (e.g. FingerCode) while other fingerprint recognition algorithms may use of a more complicated comparison function.

Biometric authentication systems are basically composed of several clients and of an authentication server. In the *enrollment* phase, a client registers by sending to the authentication server an identifier along with a reference template. Then, in the *authentication* phase, a user claiming to be a registered client sends an identifier and a fresh biometric template. The authentication server subsequently retrieves the reference template corresponding to the identifier and computes a biometric matching between the two templates. If they match, the server considers that the user is indeed who he claims to be. The inconvenient of such systems is that they store biometric templates in clear although biometrics are sensitive data: they might reveal information about the individual health, they are not revocable and some biometric data such as face images or fingerprints might be easily collected. Therefore privacy-preserving biometric authentication (PPBA) systems aims to protect the biometric templates of users from the server and from malicious outsiders. To protect against *honest but curious* servers, the authentication server is sometimes split into several parts, e.g. in [SBCS12, AAAM16]. A database thus stores the (encrypted) biometric templates, a computational server performs the biometric comparisons and an authentication server takes the decision whether the user is indeed who he claims to be or not. More details about existing PPBA, threats and challenges can be found in the comprehensive survey [PM17].

Inspired by the storage of passwords in Unix-like systems, Juels and Wattenberg [JW99] proposed a scheme to store a data computed from a biometric template that still allows for authentication. Their scheme that they call *fuzzy commitment*. Fuzzy commitments are similar to commitment schemes, where one player, called sender, first "commits" by submitting a value b in a concealed fashion to another player, called receiver. Later, the sender "opens" the committed value by proving the receiver that the value stored by the latter is a hidden version of b. Commitment schemes fulfill two basic properties: they are *concealing* in the sense that the receiver cannot guess the value of b prior to its opening by the sender; they are *binding* in the sense that the sender cannot succeed in "opening" with a value other than b. A simple analogy is

with a safe including the value b and that is locked by the sender. The receiver cannot disclose the value of b before the sender agrees to open the lock and the sender cannot substitute the value b stored in the safe with another one since the safe is kept by the receiver. Syntax and security definitions are given in Section 2.2.1. Unlike classical commitments, in fuzzy commitment schemes the value b committed by the sender and the value disclosed during the opening of the commit do not have to be exactly identical: a value close enough to the committed one also results in a successful opening. This property is achieved by leveraging error-correcting codes. Hence, fuzzy commitments are well suited to deal with the inherent noise of biometric data.

As an application of fuzzy commitments, Juels and Wattenberg propose a biometric authentication scheme, where the user sends a fuzzy commitment on his biometric template to the authentication server. The user sends his fresh biometric template and if the server obtains a successful decommitment with this biometric template, the user is authenticated. However, even if the data stored by the server does not permit to retrieve the biometric data that was used initially, the user has to send its fresh biometric template in clear to authenticate and if the authentication succeeds, the fuzzy commitment scheme construction enables the server to retrieve the biometric data that was initially concealed. Therefore, the biometric authentication system proposed by Juels and Wattenberg is not privacy-preserving.

Contribution

Going beyond Juels and Wattenberg's fuzzy commitment scheme, we design a scheme that enables to commit to a reference biometric template and later prove that there exists a freshly captured biometric template that matches the reference one *without* disclosing neither the reference nor the fresh template. The main tool for our scheme is verifiable computation (VC), a cryptographic scheme where a prover computes a proof of correctness for a given computation and provides it to a verifier who can check that the result of the computation is correct by inspecting the proof. Besides protecting the verifier against a cheating prover, several VC schemes offer additional properties. A notable one is the possibility for the prover to hide some of the inputs he provides in the computation, the hiding being obtained thanks to a zero-knowledge proof of knowledge property. The VC scheme we will leverage for our scheme are zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs) [BCCT12].

Building on the biometric commitment scheme, we design a biometric authentication protocol where the user performs self-authentication and *proves* to the authentication server that he correctly followed the process. To verify the proof of correct authentication, the user's biometric templates are not required, commitments on these templates are enough. Our biometric authentication protocol is privacy-preserving, requires little communication between the user and the authentication server and enjoys a simpler architecture than several privacy-preserving biometric authentication protocols, that usually split the authentication server into multiple parts to ensure control on data leakage.

A critical reader could be tempted to think that biometric authentication could have been addressed by simpler mechanisms. Such a solution would for instance consists of a biometric matching where the reference template is authenticated by digital signatures issued by the flight company. In this scenario, the user would present his reference template and the fresh one and a device owned by the company would verify the signature

of the template and perform the matching. This straightforward solution would have the disadvantage of revealing both the reference and the fresh templates as opposed to our solution whereby the fresh and the reference templates are not revealed. Another alternative could be envisioned that would require the user to generate a public/private key pair, to publish the public key and to prove in zero-knowledge that he indeed is in possession of the private key associated with the published public key. This solution in turn would allow some authentication but the drawback is that it would require the deployment of a heavy infrastructure to authenticate all the public keys. In contrast, our solution relies on a unique public verification key for *all users* and is thus easier to deploy.

Related work.

Our biometric commitment scheme borrows some goals to Juels and Wattenberg's fuzzy commitments [JW99], namely the scheme has to be hiding: a commitment should not reveal any information about the concealed biometric template and and strongly biding: any template that is close enough of the committed reference template should lead to a valid decommitment. However, in our scheme the decommitment is done by the user and does not require to send any template in clear to the receiver of the commit. This difference enables us to get a privacy-preserving biometric authentication scheme from the biometric commitment scheme. A drawback of our scheme is that it requires to store the reference template and some randomness that was used to conceal the reference template until the end of the authentication phase. We nonetheless stress that our biometric commitment scheme and hence our privacy-preserving biometric authentication scheme keep the convenience of biometrics: the user does not need to remember a password to authenticate, he only needs to present his biometric trait to authenticate while respecting his privacy regarding his biometric template.

Privacy-preserving biometric authentication (PPBA) protocols aims to protect information regarding the user's biometric templates from the authentication server. Several solutions have been proposed to achieve this goal. Cancellable biometrics [RCB01] schemes apply a non- invertible and repeatable transformation to a biometric template. The resulting objects is still compatible with biometric matching, often with the same comparison methods. The drawback of cancellable biometrics is the performance of the resulting authentication system: the FAR is worse than the FAR of the original system. By comparison, our PPBA scheme does not modify the underlying biometric system and the accuracy of the system is not degraded.

Another attempt is to encrypt templates but the inherent noise of biometric data prevents from computing a matching over the encrypted templates unless using homomorphic encryption. Several proposals [BCP13, BCI+07, YSK+13] leverage such encryption schemes but they are not able to perform the last step of the matching, which is a comparison to a threshold. Hence, they leak information about the result of the distance computation and attacks called hill-climbing [SBCS12] can be mounted against such systems. The strategy of hill-climbing attacks consists of recovering the reference template by successive trials, the next presented template being built by leveraging the last submitted template and the previous attempt score. Our PPBA scheme does not reveal the result of the distance computation but only the output of the authentication process and is therefore immune against such attacks. Fully homomorphic encryption [Gen09] theoretically solves the problem and enables to perform a complete matching

over encrypted data but state of the art schemes are too inefficient for a practical application. Secure multiparty computation (SMC) and more specifically two-party computation (2PC) are also relevant to achieve privacy-preserving authentication schemes. We note however that most proposals [BCF+14, HMEK11] are designed in the honest-but-curious model where the server is supposed to follow the protocol but tries to learn information about the user's data. The more restrictive but more realistic malicious model, where the server can arbitrarily deviate from the protocol, incurs computation and communication overheads: as reported in [BCF+14], even in the honest but curious model several MB of communication are required to perform an authentication. In our PPBA scheme, the communication is very low: except the commitments, only a proof is sent and zk-SNARK schemes produce very short proofs (less than 150 bytes for Groth's state of the art scheme [Gro16]).

5.3.2 Biometric Commitment Scheme

High level description

We describe hereafter our biometric commitment scheme. From a high-level view, the goal is to allow a party to commit to a biometric reference template and to later prove that another biometric template, that we will call fresh template, is close to the committed one *without* revealing any of those. We stress that both the fresh and the reference templates are known by the prover but are kept secret from the verifier, who only sees commitments on these values. As depicted in Figure 5.2, the protocol takes place be-

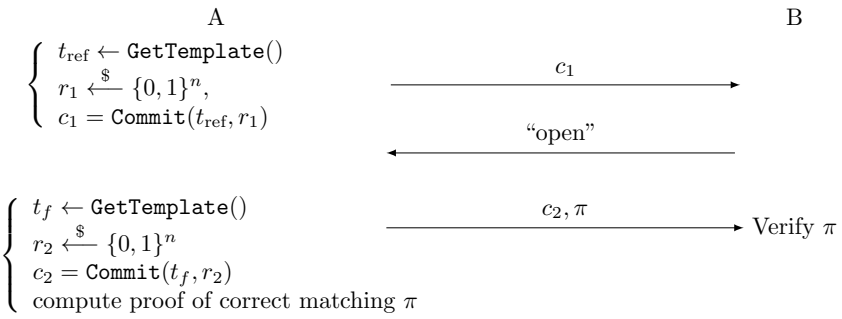


Figure 5.2 – Biometric commitment scheme high level description

tween a prover called A and a verifier called B. First, A commits to a reference biometric template t_{ref} that will be kept secret, sends that commitment to B and stores t_{ref} and the randomness involved in the commitment computing. When B wants the opening of the biometric commitment, he sends A an open request, say the message "open".Upon receiving the "open" message, A proceeds to the opening of the biometric commitment: he provides a biometric template, denoted by t_f, and computes the matching between t_f and t_{ref}. Since t_f and $t$ref are biometric templates, the matching function, which is often a distance computation, is designed such that the templates can match even if they are not identical. Since the verifier B has no control on the freshly captured biometric template, we require that the prover A commits on that fresh value and sends B the resulting commitment.

At the end of the protocol the verifier is in possession of two commitments that he cannot open. Consequently, the prover leverages the proof scheme to prove that the

commitments open to two values t_{ref} and t_f that match. The proof should therefore not reveal any information about the opened values except that i) the opening was computed following a protocol on which the verifier and the prover have agreed upon and that ii) these opened values match. A finally returns the commit on the fresh value along with the proof mentioned above to answer the open request and B checks the validity of the proof. If the proof passes, the biometric commitment opening is successful and B can assume with overwhelming probability that the fresh biometric template matches with the reference template that has been committed in the beginning of the protocol.
Some points of the previous description require further clarification. Indeed, in the opening phase of the biometric commitment, a proof system is necessary to implement the proof that there exists two templates concealed by the sent commitments and that these templates match. We assume that the related matching function will perform a distance computation and a comparison to a threshold (see Section 5.3.1 for more details on biometrics). We thus target an efficient proof scheme that is expressive enough to be able to prove that the matching has been correctly performed. The proof system must therefore be able to prove the correctness of a distance computation but also to prove a correct comparison. Moreover, the proof itself should not reveal information to the verifier about the values t_{ref} and t_f. Since the verifier only gets commitments on these values, the proof system should also prove that there exists two values t_{ref} and t_f that match and that are the openings of the given commitments. Additionally, the proof system should not only ensure that the values t_{ref} and t_f exists but also that the prover is in possession of these. Such property is called zero-knowledge proof a knowledge (zk-PoK), a formal definition is given in Section 3.3.4. As a building block meeting these requirements, verifiable computing (VC) seems very suitable: it proves that a function has been correctly computed and some VC schemes enable the prover to supply inputs in the computation about which the proof will not reveal any information and have the zk-PoK property. A notable example of such schemes are zk-SNARKs [BCCT12], see Sections 3.3.4 and 3.3 for background on zk-SNARKs.

Protocol details.

In VC schemes, a function to be verified is agreed upon the prover and the verifier. Then, receiving an input from the verifier, the prover runs the computation and produces a proof of correctness. Here, the function to be verified, denoted by f, implements the biometric matching and the commitment scheme openings. The correct execution of this function proves that the value captured after the open request phase has a distance to the committed reference value inferior to the threshold of the biometric system, denoted by τ, and that the commits sent to the verifier open to the reference and the fresh biometric templates. zk-SNARKs schemes offer the possibility for the prover to hide inputs of the computation to verify: the proof of correctness is zero-knowledge regarding these inputs. Therefore, supplying the reference value t_{ref} and of the fresh one t_f as private values in f enables to convince the verifier that they match without disclosing them. Existing practical general-purpose zk-SNARKs schemes [PHGR13, BCG$^+$13, WSR$^+$15] produce public keys to compute and verify the proof of correct computation. The evaluation and verification keys computation is heavy but has only to be run once, the key can be reused for multiple instances of the computations. Even if the verification of the proof is always efficient in these systems, the proof computation is computationally demanding [WB15]. As a consequence, carefully choosing the functions involved in the scheme

instantiation can drastically reduce the burden of the prover.
Let f be the function defined as: $f : (t_{\text{ref}}, t_f, r_1, r_2, c_1, c_2, p_k) \mapsto d_1 \times d_2 \times d_3$, with:

$$\begin{cases} d_1 & \leftarrow & \texttt{Com.Verif}(p_k, c_1, r_1, t_{\text{ref}}) \stackrel{?}{=} 1 \\ d_2 & \leftarrow & \texttt{Com.Verif}(p_k, c_2, r_2, t_f) \stackrel{?}{=} 1 \\ d_3 & \leftarrow & d(t_{\text{ref}}, t_f) \stackrel{?}{<} \tau \end{cases} \tag{5.2}$$

The function f verifies in d_1 that t_{ref} indeed opens to the commit c_1, in d_2 that t_f opens to c_2. Finally, d_3 verifies that the two templates t_f and t_{ref} match. The four values t_{ref}, t_f, r_1 and r_2 should be private inputs in the computation of f, we leverage the zero-knowledge argument of knowledge property of zk-SNARKs schemes to achieve this goal.

We assume that there exists a function `GetTemplate` that outputs a template when called. Denoting by Π the zk-SNARK scheme and by `Comm` the commitment scheme, the protocols proceeds that way:

- In the setup phase, the `Comm.KeyGen` algorithm of the commitment scheme is run to get the public key p_k later used in the commit function. Then, the Π.`KeyGen` algorithm of the zk-SNARK scheme is run and the evaluation key ek_f and the verification key vk_f are generated. The function f is defined in formula (5.2). The distance and the threshold are algorithm dependent and are set once the function is defined. The commit verification function `Comm.Verif` is also set in the circuit. The public keys p_k, ek_f and vk_f are thus supplied to A and B.

- In the commit phase, A calls `GetTemplate` and obtains a template from her biometric trait, picks a random value r_1, computes $c_1 = \texttt{Comm.Commit}(p_k, t_{\text{ref}}, r_1)$ and sends c_1 to B.

- In the open request phase, B sends a message "open" to A.

- After receiving the "open" message, A calls the `GetTemplate` function and gets a fresh biometric template t_f.

- In the opening phase, A performs a biometric matching between t_{ref} and the fresh template t_f. Then, A picks a random value r_2 and computes a commit $c_2 = \texttt{Comm.Commit}(p_k, t_f, r_2)$. Using the evaluation key ek_f, A computes a proof π of correctness for the function f defined in (5.2). In this computation, t_{ref}, t_f, $r_1$and r_2 are private inputs supplied by the prover. The correct evaluation of the function f guarantees the existence of two templates that open to the commits c_1 and c_2 and that additionally match. The argument of knowledge property of the zk-SNARK guarantees that not only such templates exists but also that the prover is in possession of t_{ref} and t_f when he computes the proof. A then sends B the proof π and the second commit c_2. Note that π contains the result of the matching since the function checks that the computed distance is inferior to the threshold. The values c_1 and c_2 are public inputs in function f and the verifier can supply them in the zk-SNARK verification algorithm Π.`Verify`.

- Using the verification key vk_f and the inputs c_1 and c_2, B checks the validity of the proof. If the proof verification passes, B can conclude that A is in possession of a fresh template t_f that matches with the initially committed template t_{ref}.

Setup:

$(ek_f, vk_f, p_k) \leftarrow \texttt{Setup}(1^\lambda, f)$,

where: $\begin{cases} (ek_f, vk_f) & \leftarrow & \Pi.\texttt{KeyGen}(f, 1^\lambda) \\ p_k & \leftarrow & \texttt{Comm.KeyGen}(1^\lambda) \end{cases}$

BioCommit:

$c_1 \leftarrow \texttt{BioCommit}(p_k, t_{\text{ref}})$, where:

- $r_1 \xleftarrow{\$} \{0,1\}^n$
- $c_1 \leftarrow \texttt{Comm.Commit}(p_k, t_{\text{ref}}, r_1)$.

OpenRequest:

- Send the message "open"

BioCommitOpen:

$(c_2, \pi) \leftarrow \texttt{BioCommitOpen}(ek_f, p_k, t_f, t_{\text{ref}}, c_1, r_1)$ where:

- $r_2 \xleftarrow{\$} \{0,1\}^n$
- $c_2 \leftarrow \texttt{Comm.Commit}(p_k, t_f, r_2)$
- $\pi \leftarrow \Pi.\texttt{Prove}(ek_f, t_{\text{ref}}, t_f, r_1, r_2, c_1, c_2)$.

Verify:

$out \leftarrow \texttt{Verify}(vk_f, p_k, \pi, c_1, c_2)$,

where: $out = \Pi.\texttt{Verify}(vk_f, p_k, \pi, c_1, c_2)$.

Figure 5.3 – Biometric commitment syntax

Let f be the function defined above in formula (5.2), the syntax of the protocol is given in Fig. 5.3.

Security

The security properties required for our scheme are related to the security properties of Juels and Wattenberg fuzzy commitment scheme [JW99], namely the scheme has to be *correct*, *hiding* and *strongly binding*. The correctness guarantees that a legitimate user that follows honestly the protocol will always pass the verification. The hiding property protects the user's templates from the server by asserting that the commitment he sends do not provide any useful information to the server regarding his template. Finally, the strong biding property states that the commitment sent in the first part of the protocol (the `BioCommit` phase) cannot be opened successfully unless the user possesses a biometric template that is close enough to the committed reference template (more precisely that belongs to the ball of radius τ, the system threshold, centered in the reference template). In the following, we keep the notations of Section 5.3.2 and we assume the existence of an algorithm `GetTemplate` that outputs a biometric template when called.

Correct. Assume that A ran the `BioCommit` algorithm honestly and received an "open" message from B. The scheme is correct if `Verify` returns 1 each time A runs the `BioCommitOpen` algorithm honestly with a biometric template whose distance from the reference one is inferior to the threshold.
More formally, let denote by τ the threshold of the biometric system, by t_{ref} the reference template of A and by t_f a fresh template such that: $d(t_f, t_{\text{ref}}) < \tau$. Suppose that A runs the `BioCommit` algorithm and gets a commit c_1. He then receives the "open" message from B and runs the `BioCommitOpen` algorithm to get a proof π and a commit c_2. The scheme is *correct* if: $\texttt{Verify}(vk_f, p_k, \pi, c_1, c_2) = 1$.

Hiding. The scheme is hiding if no useful information about A's templates can be extracted from the information possessed by the receiver B. This is formalized by two attack games `hide_attack1` and `hide_attack2`, in which the attacker first sees the two commitments and the proof computed by the sender and then tries to distinguish between two possible pairs of reference (resp. fresh) templates. Let $\mathcal{A}$ be a PPT adversary, the hiding properties are formalized by the mean of the following games:

`hide_attack_1`(1^λ)

$(ek_f, vk_f, p_k) \leftarrow \texttt{Setup}(1^\lambda, f)$
$(t_0, t_1) \leftarrow \texttt{GetTemplate}()$
with $t_0 \neq t_1$
$d \xleftarrow{\$} \{0, 1\}$
$c_1 \leftarrow \texttt{BioCommit}(p_k, t_d)$
$t_f \leftarrow \texttt{GetTemplate}()$ with $d(t_f, t_d) < \tau$
$(c_2, \pi) \leftarrow \texttt{BioCommitOpen}(p_k, ek_f, t_d, t_f, c_1, r_1)$
$d^* \leftarrow \mathcal{A}(c_1, c_2, \pi, t_0, t_1)$
if $d = d^*$:
  return 1,
otherwise:
  return 0

`hide_attack_2`(1^λ)

$(ek_f, vk_f, p_k) \leftarrow \texttt{Setup}(1^\lambda, f)$
$t_{\text{ref}} \leftarrow \texttt{GetTemplate}()$
$c_1 \leftarrow \texttt{BioCommit}(p_k, t_{\text{ref}})$
$t_{f,1} \leftarrow \texttt{GetTemplate}()$ with $d(t_{\text{ref}}, t_{f,1}) < \tau$
$t_{f,2} \leftarrow \texttt{GetTemplate}()$ with $d(t_{\text{ref}}, t_{f,2}) < \tau$
$d \xleftarrow{\$} \{0, 1\}$
$(c_2, \pi) \leftarrow \texttt{BioCommitOpen}(p_k, ek_f, t_{\text{ref}}, t_{f,d}, c_1, r_1)$
$d^* \leftarrow \mathcal{A}(c_1, c_2, \pi, t_{f,1}, t_{f,2})$
if $d = d^*$:
  return 1,
otherwise:
  return 0

The scheme is hiding if for every PPT adversary and for $i = 1, 2$:

$$\Pr\left[\texttt{hide\_attack\_i}(1^\lambda) = 1\right] \leqslant \frac{1}{2} + negl(\lambda)$$

Strongly Binding. The strongly biding property captures a notion similar to the biding property of commitment schemes in the case of biometric commitments that deal with inherently variable data. A scheme is strongly binding if the biometric commitment cannot be opened successfully by the sender A except if the template captured after the open request belongs to the ball centered in t_{ref} of radius τ. This is formalized by the `bind_attack` game given below. In the `bind_attack` game, the adversary first commits to a reference value t_{ref} and gets a commit c_1. Then it generates a value t_f that is out of the ball centered in t_{ref} of radius τ. The adversary wins if it is able to compute a proof π and a commit c_2 such that the `Verify` algorithm, on inputs π, c_1 and c_2 returns 1. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ denote such an adversary.

`bind_attack`(1^λ)
$(p_k, ek_f, vk_f) \leftarrow$ `Setup`$(1^\lambda, f)$
$t_{\text{ref}} \leftarrow$ `GetTemplate`()
$c_1 \leftarrow \mathcal{A}_1(p_k, 1^\lambda, t_{\text{ref}})$
$t_f \leftarrow$ `GetTemplate`() with $d(t_{\text{ref}}, t_f) > \tau$
$(\pi, c_2) \leftarrow \mathcal{A}_2(p_k, 1^\lambda, t_{\text{ref}}, c_1, t_f)$
if `Verify`$(vk_{\mathcal{C}}, p_k, \pi, c_1, c_2) = 1$:
  return 1
else:
  return 0

A biometric commitment scheme is binding if for every PPT adversary $\mathcal{A}$:

$$\Pr\left[\texttt{bind\_attack}(1^\lambda) = 1\right] \leqslant negl(\lambda)$$

The security of the scheme is defined as follows,

Definition 16. *The biometric commitment scheme is secure if it is correct, hiding and strongly biding.*

And we have the following result:

Theorem 9. *The biometric commitment is secure if the underlying commitment scheme is secure and if the zk-SNARK scheme is secure.*

Proof. The correctness of the biometric commitment scheme is a direct consequence of the correctness of the commitment scheme and of the zk-SNARK scheme. □

Proof. We prove that the scheme is **hiding** with a sequence of games.

Game 0 is the `hide_attack_1` game.

Game 1 is the same as game 0, except that the result c_1 of `BioCommit` is replaced by a random value. The hiding property of the underlying commitment scheme ensures that the adversary has negligible advantage to be able to discriminate between a legitimate commitment on t_{ref} and a random value.

Game 2 is the same as game 1, except that the commitment c_2 is also replaced by a random value. The same argument that the one given in game 1 holds: the advantage in distinguishing a random value from a commitment is negligible.

Game 3 is the same as game 2 except that the user makes a call to the zk-SNARK simulator to replace the proof *pi* by a random value that will still be accepted by the zk-SNARK verification algorithm. The zero-knowledge property of zk-SNARK schemes ensures that the advantage of an adversary to distinguish that random string from a legitimate proof is negligible.

The probability that $\mathcal{A}$ wins game 3 is now $1/2$ because all the information the adversary receives is random.
Since the transition between each game being negligible, the probability of winning game 0 is therefore: $1/2 + negl(\lambda)$, where $negl(\lambda)$ is the sum of the negligible functions involved in the transitions. The same arguments apply for `hide_attack_2`. □

Proof. We prove that the scheme is **strongly biding** by directly evaluating the winning probability of an adversary. To win the `bind_attack` game, the adversary should be able to forge a proof that $d(t_{\text{ref}}, t_f) < \tau$. Since the proof system verifies that the distance is indeed inferior to the threshold and also that the templates are concealed in the commitments passed to the verification algorithm, the adversary should either be able to forge a proof of a false statement or to open a commitment to a wrong value. Since the zk-SNARK scheme is secure and therefore sound, it is impossible except with negligible probability to forge a proof of a false statement. Since the commitment scheme is secure and hence biding, it is impossible except with negligible probability to open a commit to a wrong value. The probability that a PPT adversary wins the `bind_attack` game is the sum of the two previous probabilities and is therefore negligible. □

5.3.3 Privacy-preserving Biometric Authentication Protocol

Leveraging the biometric commitment scheme we described in Section 5.3.2, we build a privacy-preserving biometric authentication (PPBA) protocol. Using the biometric commitment protocol as a building block for a PPBA scheme is not straightforward and several challenges remain. Indeed applying the commitment scheme with a user as the commitment sender and an authentication server as the commitment receiver provides guarantees to the server that a user is in possession of two biometric templates that match and that correspond to the commitments that it received. However, since the server never sees the reference template, it has no guarantees on that value and a malicious user could commit to a pathological template such as e.g. a vector with zeros for each component. Moreover, the biometric commitment gives no guarantees that the second biometric template has been freshly captured, i.e. after the open request sent by the server. We tackle the reference template issue by enjoining the user to extract his template and to commit on it in front of a trusted party that signs the commit to authenticate it. This not only ensures that the reference biometric template has been extracted by means of a trusted sensor but also that the commit on the template is tied to a legitimate user. We thwart replay attacks by requiring that, in the open request phase of the biometric commitment protocol, the server sends a nonce that the user must integrate in the computation of the second commitment and therefore in the proof computation. This requires a slight modification of the function f that was defined in formula (5.2) in Section 5.3.2 and for which the zk-SNARK scheme computes a proof of correctness: we need to add some constraints to ensure that the nonce sent by the server is indeed integrated in the second commitment computation. In detail the new function f that implements the verification is now:

$f : (t_{\text{ref}}, t_f, r_1, r_2, c_1, c_2, p_k, N_B) \mapsto d_1 \times d_2 \times d_3 \times d_4$, with:

$$\begin{cases} d_1 & \leftarrow & \texttt{Com.Verif}(p_k, c_1, r_1, t_{\text{ref}}) \stackrel{?}{=} 1 \\ d_2 & \leftarrow & R \stackrel{?}{=} r_2 \parallel N_B \\ d_3 & \leftarrow & \texttt{Com.Verif}(p_k, c_2, R, t_f) \stackrel{?}{=} 1 \\ d_4 & \leftarrow & d(t_{\text{ref}}, t_f) \stackrel{?}{<} \tau \end{cases} \tag{5.3}$$

Here the intermediate constraint $R \stackrel{?}{=} r_2 \parallel N_B$ is added to check that the nonce provided by the authentication server to the user has indeed been supplied to compute the commitment on the fresh template. The protocol is fully described below, it takes places between a trusted party, denoted by TP, an authentication server, denoted by S, and a user denoted by U_i. The zk-SNARK algorithm is denoted by Π and the commitment algorithm is denoted by `Comm`. The signature algorithm is denoted by Sgn.

Setup

- TP runs the setup algorithm of the biometric commitment protocol and the key generation algorithm of the signature scheme:

$$\begin{aligned} (ek_f, vk_f) &\leftarrow \Pi.\texttt{KeyGen}(1^\lambda, f), \\ cp_k &\leftarrow \texttt{Comm.KeyGen}(), \\ (p_k, s_k) &\leftarrow \texttt{Sgn.KeyGen}(1^\lambda) \end{aligned}$$

- TP distributes the keys to the participants:
  - S receives vk_f, cp_k, p_k.
  - U_i receives ek_f, cp_k.

Enrollment

- U_i's biometric template $t_{\text{ref},i}$ is extracted by the mean of a sensor that belongs to TP and U_i picks an identifier id_i.
- TP runs the `BioCommit` algorithm, gets the commitment $c_{1,i}$ and the randomness $r_{1,i}$, signs the pair (commit,id) by running $\sigma \leftarrow \texttt{Sgn.Sign}((c_{1,i}, id_i), s_k)$.
- TP gives $c_{1,i}, id_i, r_{1,i}, \sigma$ to U_i.
- U_i sends $(c_{1,i}, id_i, \sigma)$ to S.
- S verifies the signature σ: $\texttt{Sgn.Verif}((c_{1,i}, id_i), \sigma, p_k)$. If the verification passes, S stores the pair $(c_{1,i}, id_i)$ in its database.

Authentication

- U_i sends S his claimed identifier id_i.
- S generates a nonce $N_{B,i} \in \{0,1\}^\ell$ and sends it to U_i.
- Using his own device, U_i extracts a fresh template $t_{f,i}$.

- U_i runs the `BioCommitOpen` algorithm with inputs: $c_{1,i}$,$r_{1,i}$, $t_{\text{ref},i}$, $t_{f,i}$. During the execution of the algorithm, U_i picks a random value $r_{2,i} \xleftarrow{\$} \{0,1\}^{n-\ell}$ and concatenates it with $N_{B,i}$ to compute the commitment $c_{2,i}$ on $t_{f,i}$. He also gets a proof π_i and sends $(c_{2,i}, \pi_i)$ to S.

Verification

- S runs the biometric commitment `Verify` algorithm with inputs $N_{B,i}$, $c_{1,i}$,$c_{2,i}$ and π_i.
  If the verification passes, U_i is correctly authenticated, otherwise the server returns $\perp$.

Security Goals

We adapt the security goals of Abidin et al. [AAAM16] to our scheme by requiring our PPBA scheme to be *correct* and *private*.

Correctness: for all enrolled user under the identity *id* with a corresponding template t_{ref}, the authentication phase with a fresh template t_f is successful if and only if $d(t_{\text{ref}}, t_f) < \tau$.

Privacy: we define the attack game `private_attack` below, which basically models a malicious server that would run the authentication process and try to distinguish between two pairs of templates. In this game, a PPT adversary tries to distinguish between two pairs of matching templates he has chosen after being given the outputs of the protocol, which are the two commitments and the proof. Let $\mathcal{A}$ be a PPT adversary. Here, the `Enroll` algorithm represents the enrollment process as run by the trusted party TP. This algorithm therefore outputs a commitment on the reference value t_{ref} and its signature.

$\underline{\texttt{private\_attack}(1^\lambda)}$
$(p_k, ek_{\mathcal{C}}, vk_{\mathcal{C}}) \leftarrow \texttt{Setup}(1^\lambda, f)$
$t_{\text{ref}_1}, t_{f_1}, t_{\text{ref}_2}, t_{f_2} \leftarrow \mathcal{A}(p_k, ek_f, vk_f, 1^\lambda)$
with $d(t_{\text{ref}_1}, t_{f_1}) < \tau$ and $d(t_{\text{ref}_2}, t_{f_2}) < \tau$
$b \xleftarrow{\$} \{0,1\}$
$(c_1, \sigma) \leftarrow \texttt{Enroll}(t_{\text{ref}_b})$
$(c_2, \pi) \leftarrow \texttt{Authenticate}(ek_f, p_k, t_{f_b}, t_{\text{ref}_b}, c_1, r_1, N_B)$
$b^* \leftarrow \mathcal{A}(c_1, \sigma, c_2, \pi, t_{\text{ref}_1}, t_{f_1}, t_{\text{ref}_2}, t_{f_2}, N_B)$
If $b = b^*$, return 1
else return 0

The scheme if private if for every PPT adversary:

$$\Pr\left[\texttt{private\_attack}(1^\lambda) = 1\right] \leqslant \frac{1}{2} + negl(\lambda)$$

Theorem 10. *The PPBA scheme is private and correct if the signature scheme, the commitment scheme and the zk-SNARK scheme are secure.*

Proof. The *correctness* of the PPBA scheme is a consequence of the correctness of the biometric commitment scheme: a legitimate user, enrolled under identity id with the reference template t_{ref} that honestly authenticates with a fresh template t_f will run the `BioCommitOpen` algorithm with the nonce N_B provided by the server. The correctness of the Biometric Commitment scheme guarantees that the proof will be accepted in the verification phase and therefore that the user will be successfully authenticated.
We prove the *privacy* of our PPBA scheme with a sequence of games.

Game 0 is the `private_attack` game.

Game 1 is the same as game 0 except that the result c_1 of `BioCommit` is replaced by a random value and a signature of that random value is asked to the trusted party. Since the commitment scheme is hiding, the advantage in distinguishing a commitment from a random value is negligible. The signature of the replaced value is still valid.

Game 2 is the same as game 1 except that in `Authenticate`, the commitment c_2 is replaced by a random value. Again, the hiding property of the commitment scheme gives a negligible advantage to the adversary in distinguishing a random value from a commitment.

Game 3 is the same as game 2 but the proof is replaced by a random string generated by the simulator of the zk-SNARK scheme, that random string being accepted by the verification algorithm. The zero-knowledge property of the zk-SNARK scheme gives negligible advantage to an adversary in distinguishing a random proof from a legitimate one.

From now, the probability that $\mathcal{A}$ wins game 3 is $1/2$ because all the information it gets excepts the pairs of template is random. Since all the transition between the games have negligible probabilities, the probability to win `private_attack` is therefore $1/2 + negl(\lambda)$. □

5.3.4 PPBA Protocol Instantiation

To instantiate the protocol described in Section 5.3.3, we need to specify a zk-SNARK scheme and a commitment scheme. The choice of the commitment scheme is crucial since the scheme is somewhat embedded in the verifiable computation scheme: the function f specified in equation (5.3) computes a commitment from a biometric template and the proof that the user joins to the commitments is a proof of correct execution for f. All the zk-SNARK schemes based on Gennarro et al.'s quadratic arithmetic programs [GGPR13] produce a proof of size independent of the computation to verify. The proof verification is fast but the drawback is the prover's work which is the bottleneck of such schemes. The efficiency of the prover is quasi-linear in the size of the arithmetic circuit (see Section 3.3) that represents the function to verify. More precisely, the efficiency is quasi-linear in the number of multiplicative gates of the arithmetic circuit, the addition gates are free in this setting because their contribution is taken into account in the next multiplicative gate of the circuit. We choose Groth's state of the art zk-SNARK scheme [Gro16] because it is the most efficient of existing schemes and besides, it produces the shorter proof of zk-SNARK protocols.

We first assume that the biometric face recognition system is Schroff et al.'s Facenet [SKP15]. This state of the art convolutional neural network algorithm takes as input

a face image and outputs a vector with 128 floating-point numbers components. To decide if two vectors come from the same individual an Euclidean distance is computed. Since minimizing the size of the circuit that represent the function to verify improves the performance of the zk-SNARK scheme, we implement a squared euclidean distance to avoid an expensive square root computation. We also convert the floating-point numbers into a fixed-point representation on 32 bit integers, the resulting biometric template is thus 4096 bit long. We study the impact of such transformation in Section 5.3.5. Regarding the commitment scheme, we choose Kawachi et al.'s commitment scheme [KTX08], which is built from Ajtai hash function [Ajt96]. Ajtai hash function is provably secure and is efficiently implementable in the zk-SNARK efficiency model [KZM+15] (recall that a computation to be verified has to be expressed as a circuit over a finite field and that addition in this circuit are free). Kawachi et al. prove that the resulting commitment scheme is statistically binding and computationally binding. Let q be the prime number equal to the cardinality of the prime field where the computations of the zk-SNARK scheme take place. Let (n, m) be two integers such that: $m > n \log q$. We denote the Ajtai hash function family, that has been defined in Section 2.2.2, by $f_A(x) = \mathbf{A} \cdot x \mod q$, where $\mathbf{A} \in \mathcal{M}_{n,m}(\mathbb{F}_q)$. Let B and C be two randomly chosen (n, m)-matrices in $\mathcal{M}_{n,m}(\mathbb{F}_q)$. Denoting by A the matrix defined by $A = [B\ C]$ and by ρ a randomly chosen value in $\{0,1\}^m$, the commitment of an input string $s \in \{0,1\}^m$ is thus:

$$Com_{\mathbf{A}}(s; \rho) := f_{\mathbf{B}}(\rho) + f_{\mathbf{C}}(s) \tag{5.4}$$

Using Merkle-Damgård construction, Kawachi et al. extend this commitment scheme to arbitrary length strings in [KTX08]. Based on Kosba et al. [KZM+15], we choose q to be the 254-bit prime where arithmetic circuits are defined, $m = 1524$ and $n = 3$.

5.3.5 Experimental Results

On the impact of using integer templates instead of float templates

Most of the VC systems deal with computations that are described over a large finite field. This enables to naturally emulate computations over integers as defined in the majority of programming languages, the extension to signed integers being also natural. Setty et al. [SVP+12] design circuits that allow for floating-point rational computations at the expanse of overhead costs in proving time. State of the art biometric feature extraction algorithms perform operations over floating point numbers to turn a biometric trait into a template. This enters in conflict with VC schemes that can only deal efficiently with integers. Even if there were some trials to design feature extraction algorithms that only deal with integers, the accuracy of the latter is not satisfying compared to the state of the art (see for example [WLCS18] and the reference therein). Hence a natural question arises: what is the impact on a biometric system accuracy to convert templates with floating point numbers into templates with integers by scaling and truncating each floating point value of the same factor ?

Using a face database of our laboratory and a feature extraction algorithm, we compared the accuracy between the floating point numbers templates and the integers templates obtained by scaling. Note that we kept the threshold of the original system and scaled it according to the scaling factor. The results are shown in Table 5.4. We conclude that the use of integers does not significantly change the accuracy of the biometric system and that our method could be used in real life.

	Float templates	Int templates (scale = 10^3)	Int templates (scale = 10^4)	Int templates (scale = 10^5)
FAR (%)	0.1	0.1	0.1	0.1
FRR (%)	4.3	5	4.3	4.3

Table 5.4 – Benchmarks between templates with decimal values and scaled and truncated integer templates deduced from the original ones

Biometric Commitment Implementation

We now report some figures about the implementation of our biometric commitment scheme. The experiments were performed on a 8-core machine running at 2.9 GHz with 16 GB of RAM, using no parallelization. The verifiable computation system is Groth's state of the art zk-SNARK [Gro16] and its implementation within the `libsnark` library. [1] Afterwards, we also experiment the scalability of our scheme by letting the number of components of the templates increase.

Template component numbers	EK size	VK size	`Keygen`	`Prove`	`Verify`	Proof size
128	4.5 MB	61 kB	1.91 s	0.66 s	0.0015 s	120 B
256	8.2 MB	61 kB	3.37 s	1.32 s	0.0015 s	120 B
384	11.7 MB	61 kB	4.90 s	1.75 s	0.0015 s	120 B
512	15.9 MB	61 kB	6.46 s	2.57 s	0.0015 s	120 B

Table 5.5 – Experimental results on the Biometric Commitment Scheme.

Table 5.5 reports benchmarks on the implementation of the Biometric Commitment Scheme proving part, i.e. the proof that two templates match and that they are concealed within the two given commitments. Note that the key generation algorithm, besides being efficient, has only to be run once: the evaluation and verification keys can be used for every biometric commitments. The proving time is low enough to imagine the integration within a customized boarding gate. The next section details a scenario where our privacy-preserving commitment scheme is relevant.

5.3.6 Use case: privacy-preserving boarding check

Let us consider a scenario where a flight company, denoted by C, proposes a privacy-preserving boarding check for its flights, leveraging our privacy-preserving biometric authentication protocol. We assume that users who want to benefit of this scheme are in possession of a smartphone, which is equipped with a face recognition algorithm. Any user A in possession of a smartphone will proceed in two steps to board. First, a trusted party, e.g. the International Civil Aviation Organization (ICAO), runs the setup algorithm of the privacy-preserving authentication algorithm. Then, at the enrollment phase, user A extracts a template from her face by means of a sensor that belongs to the trusted party. As in Section 5.3.3, the trusted party computes the commitment on the reference template and signs it. Since many countries require flight companies to provide personal information about the passengers they carry, the trusted party can perform some background checks on user A before signing the commitment, for instance

[1] Libsnark, a C++ library for zkSNARK proofs, available at `https://github.com/scipr-lab/libsnark`

that user A does not belong to a blacklist. A then gets the reference template, the randomness used in the process and the signed commitment. A can then sends the signed commitment to the flight company that first verifies the validity of the signature. If the signature verification passes, the flight company builds a flight ticket containing the commit and additional information such as the flight number, the seat number and the validity duration of the ticket. This ticket is signed by the flight company and sent back to A. In a second phase, when A arrives at the airport, she enters in a specific boarding corridor equipped with a device that embeds the verification keys of the zk-SNARK algorithm and signature verification key of the flight company. The device is assumed to be able to communicate with A's device. It first generates a nonce and sends it to A's smartphone. Using her smartphone, A extracts her face biometric template, performs the matching and proves it was correctly done, integrating the nonce as described in Section 5.3.3. She presents the proof, the freshly computed commitment and the ticket to the device, which checks the authenticity and the validity of the ticket and verifies the proof, with inputs the commitments and the nonce. If they all pass, the corridor opens and A can board the airplane. Because the template extraction happens in a closed corridor, A cannot authenticate and then give her smartphone with the commitment and the proof of correct authentication to someone else. We assume that the user's mobile is able to perform liveness detection to detect spoofing attacks [MNL14]. Hence, A cannot be impersonated by an individual that would present a picture of A's face in front of the smartphone camera. The experiments reported in Section 5.3.5 show that the boarding corridor we envision is of practical interest. Indeed, the measurements reported in Table 5.5 show that the time for the user to produce a proof that the reference and fresh templates open to the given commitments and match will last no more than 2.5 seconds. This time can even be inferior to one second if the template has 128 components. At the end of the corridor, the user can finally present the proof and the signed ticket. The device only has to embed the verification keys of the zk-SNARK and the signature schemes. These keys are small enough (61 kB for the zk-SNARK scheme, 265 bits to 4 kB for the signature scheme) to fit in an embedded device. The device can then verify the authenticity of the ticket and the validity of the proof. The verification is really quick and should not represent any bottleneck in the process.

5.4 Chapter's Conclusion

In this chapter, we presented two schemes that build on the properties of zk-SNARKs. Both of these schemes leverage the commit-and-prove paradigm by first requiring the prover to commit on a data that should be kept secret and then proving some property that is inherent to this committed data. The two proposed schemes do not depend on the choice of the underlying zk-SNARK scheme but rather rely on the succinctness and zero-knowledge argument of knowledge properties. Except for these two properties, the only requirement is that the scheme is expressive enough to be able to produce a proof of a correct commitment computation. We chose the Ajtai hash function to implement the commitments because this hash function is well suited for computation expressed as arithmetic circuits. Choosing the underlying zk-SNARK scheme to implement the protocols leads to variations in the efficiency of the prover (and the key generation algorithm) and in the proof size that is succinct but whose size can be slightly different depending on the selected scheme. As a consequence, any improvement on the state of the art zk-SNARK schemes would lead to an improvement of the proof size and of

the schemes' efficiency. We implemented the two proposed schemes using the `libsnark` library and the experiments we ran showed that, using these schemes, the prover's running time is adapted to a practical use.

Conclusion

In this thesis, we studied verifiable computation (VC) along two different directions. A first direction adopts a theoretical point of view: starting from the conflict between the expressiveness and the efficiency that is inherent to all existing VC systems, we proposed a new VC system that moves the trade-off point toward efficiency while keeping the original expressiveness. We achieved this by leveraging proof composition between an outer VC scheme that is general-purpose and an inner VC scheme that is specialized. The outer scheme provides expressiveness while the inner scheme provides efficiency each time the operation for which it is designed is called in the computation to verify. As acknowledged by experimental results we measured, our contribution brings verifiable computation closer to practicality: compared to a matrix multiplication operation verified by a general purpose scheme, we report a proving time improvement by a factor of 10. We were also able to implement a verifiable neural network example and to verify it for parameter sizes that could not be reached if the same network is verified in the general purpose VC scheme. Additionally, our scheme allows for the composition of several different inner VC schemes. In our contribution we focused on the matrix multiplication problem but other specialized and efficient VC schemes could fit our embedded proof framework. Thanks to our proof composition technique, using efficient sub-protocols suited for each class of computation, complex sequences of computations can be verified in a single VC system. Our contribution therefore paves the way to a new modular design of practical oriented VC schemes.

The second direction of research adopted by this thesis leverages on privacy properties of existing VC schemes along a practical approach. Several VC schemes provide a zero-knowledge property for their resulting proof and the expressiveness of such schemes enables to build practical zero-knowledge proofs for predicates that were previously impossible to prove in practical time, such as knowledge of hash function pre-images. Moreover, the proof is short and the associated cryptographic guarantees make it a good candidate as a token for authentication. Hence, we designed a scheme that enables to modify an authenticated document while keeping some authenticity on the final document by adding a proof that no unauthorized modification has been performed. The scheme is close to redactable signatures but is more space efficient, especially when the document is an image for which each pixel could be redacted. This space efficiency is obtained by the succinctness of zk-SNARK schemes, the VC scheme we consider generates a proof of constant size (less than 300 bytes) regardless of the computation under verification. We also proposed a biometric authentication scheme where, thanks to the zero-knowledge property, the privacy of the biometric templates involved in the authentication is preserved. The overall idea of the scheme is to let the user perform self-authentication and to prove to the authentication server that the process was performed correctly. The scheme builds on the commit-and-prove paradigm to achieve privacy of

the templates based on a model inspired by Juels and Wattenberg's fuzzy commitments [JW99]. Those two applications highlight the capabilities of existing VC schemes beyond the basic setting of computational verifiability by managing to meet two conflicting requirements, that is, authentication and privacy.

About verifiable biometric matching

The initial focus of this thesis was to study how verifiable computation could bring confidence in the biometric matching process. Biometric matching includes liveness detection, feature extraction, distance computation and comparison to a threshold, therefore, each of these operations have to be verified to get a verifiable biometric matching and we made several steps toward this goal. The distance computation and the comparison to a threshold can be verified with the state of the art VC schemes and we implemented them. Moreover, leveraging the zero-knowledge property of zk-SNARK schemes, not only did we manage to achieve a verifiable distance computation and comparison to a threshold but also our scheme preserves the privacy of the biometric templates involved in the matching process. We note that unlike distance comparison and threshold comparison operations, liveness detection and feature extraction algorithms are way more complex to verify and several obstacles remain mainly for two reasons:

- The current VC schemes that are expressive enough to verify feature extraction algorithms have limitation in terms of the size of the computation they can deal with. Wahby et al. [WSR+15] report experiments showing that existing implemented QAP-based VC systems have a limited gate budget: such systems are not able to verify a function whose representation as an arithmetic circuit has more than ten million gates. As a consequence, complex functions that come from signal processing and that involve large computations require either a new and efficient circuit representation or significant improvements in the efficiency of the overall VC schemes.

- Feature extraction algorithms inherently deal with floating point numbers while VC schemes verify computations defined over a finite field: designing neural networks that only perform computations over integers is an active research area [WLCS18] but for now, the accuracy of the resulting schemes is not as good as the accuracy of state of the art schemes. And verifying floating-point operations incurs non-negligible overheads, as measured by Setty et al. [SVP+12].

The latter points highlight that significant progress is required to achieve a practical verifiable feature extraction scheme. Our contribution on proof composition makes a step toward this goal because such algorithms involve large matrix multiplications and our embedded proof scheme improves the proving time of computations that include matrix multiplications. In addition, our embedded proof scheme is flexible enough to integrate further advances in VC schemes, especially because any improvement on the general VC scheme would help to decrease the proving time since the protocol design is agnostic of the underlying GVC scheme. Another improvement possibility would be to select a specialized matrix multiplication VC scheme that would be more efficient than the one based on Thaler's specialization of the sumcheck protocol [Tha13]. A candidate is Freivalds's matrix multiplication verification scheme [Fre77], which could lead to a very efficient embedding, the drawback is that this scheme requires verifiable

randomness. We explain the reason of this requirement and propose a possible solution as a future work in the next section.

Future directions

Embedded Proofs

In our embedded proof contribution, we embedded the sumcheck protocol in a general purpose VC scheme. We leveraged the sumcheck protocol to verify matrix multiplication, as described by Thaler. We note that the sumcheck protocol is initially a protocol that efficiently checks the evaluation of a multivariate polynomial. As mentioned above, Freivalds' algorithm is another good candidate as a specialized an efficient matrix multiplication verification scheme. In Freivalds' protocol, to check that a $n \times n$ matrix C is indeed the result of the product of two $n \times n$ matrices A and B, the proof is simply the matrix C. The verifier then selects a random vector x and performs the comparison between the matrix-vector products $A \cdot (B \cdot x)$ and $C \cdot x$. If the products are equal, the verifier accepts C as the result of $A \times B$, otherwise he rejects. The probability that the verifier accepts a false product is inferior to $1/2$. To decrease the soundness error, the verifier can repeat the process several times. The scheme is very efficient because three matrix-vector products are enough to check the correctness of the result for a cost of $O(n^2)$ operations while performing the matrix product costs $O(n^3)$ operations. The drawback of this scheme is that it requires a lot of randomness: to get a 2^{-k} soundness error, nk random values are required. Hence, to embed Freivalds' scheme into a general VC scheme there is a need for verifiable randomness. Apart from matrix multiplication algorithms, different specialized protocols that deal with other kind of operations and that embed well in VC schemes remain to be found. Besides, it could be interesting to implement such sub-protocols inside the open-source `libsnark` library, concretely realizing the modular design we sketched in the beginning of the conclusion.

Commit and prove

In the commit and prove schemes we designed in our contributions, the decommitment phase is implemented inside the VC scheme: the function for which a proof of correctness is produced recomputes the commitment. The efficiency metric in QAP-based VC schemes is directly linked to the number of multiplicative gates of the circuit that implements the function to verify. Therefore, some commitment schemes that are not considered efficient in unverified schemes can be particularly relevant regarding efficiency when implemented in VC schemes. The commitments we consider in our contributions are built from the Ajtai hash function, which has an efficient circuit representation. A possible improvement would be to replace such commitments by Pedersen commitments, implemented with an elliptic curve. This curve would be defined over the underlying field of the zk-SNARK scheme and its point multiplication implemented in the arithmetic circuit from which the QAP is defined. Kosba et al. [KZM+15] proposed and implemented such elliptic curve, the circuit they obtain has about 6 multiplicative gates per bit of the multiplication scalar. Bowe et al. [BGM17] later improve the latter result, obtaining about 4.2 multiplicative gates per bit of the scalar. Such commitments are not only efficient but also rely on a well studied cryptographic assumption, namely the discrete logarithm assumption.

Verifiable randomness

We already mentioned Freivalds' algorithm as a possible efficient matrix multiplication verification algorithm that could be embedded inside a general VC scheme based on our technique. We also mentioned that the drawback of this algorithm is that it requires randomness and hence the new requirement for verifiable randomness to be embedded in a VC scheme. We envision two possible directions for providing such randomness: the first approach is external and would leverage verifiable random functions [MRV99] to provide a source of randomness for the prover that the verifier could verify. However, the drawback is that such schemes require a secret key for verification and the general VC scheme relying on such verifiable random function would no longer keep its public verifiability property. Another direction is internal and would leverage an elliptic curve point multiplication implemented inside the circuit of the general VC scheme. The idea would be to build on the scheme suggested by Chevalier et al. [CFPZ09] who prove that the least signifiant bits of a random point abscissa of an elliptic curve is indistinguishable from an uniform bit-string. Hence, after receiving a seed from the verifier, which would be the scalar in the elliptic curve point multiplication, the prover could verifiably extract randomness by computing an elliptic curve point multiplication and verifiably selecting bits of the resulting point abscissa.

Linear Interactive Proof Framework

QAPs are very efficient objects to check correct circuit evaluation. Bitansky et al. gave a framework that abstracts the way to obtain a secure protocol from the QAP objects, namely the linear interactive proof (LIP) framework. In this framework, Groth's zk-SNARK scheme is quasi-optimal: there are 3 group elements in the proof while a proof with one group element is impossible. Hence, new ideas are required to go beyond the LIP framework. A possible idea would be to leverage two-party computation to perform the polynomial divisibility check in QAPs that proves circuit correctness. Firstly, the prover-verifier setting fits well the two-party computation setting. Moreover, the security is information-theoretic and two-party computation is very efficient because it mostly rely on symmetric key cryptography while the LIP framework requires an homomorphic asymmetric scheme.

Bibliography

[2dd] 2D-Doc. `https://ants.gouv.fr/Les-solutions/2D-Doc`. Accessed: 2017-01-10.

[AAAM16] Aysajan Abidin, Abdelrahaman Aly, Enrique Argones-Rúa, and Aikaterini Mitrokotsa. Efficient verifiable computation of XOR for biometric authentication. In *Cryptology and Network Security - 15th International Conference, CANS 2016, Milan, Italy, November 14-16, 2016, Proceedings*, pages 284–298, 2016.

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[ACK+02] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. Seti@home: an experiment in public-resource computing. *Commun. ACM*, 45(11):56–61, 2002.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 99–108, 1996.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 56–73, 2004.

[BBD+10] Christina Brzuska, Heike Busch, Özgür Dagdelen, Marc Fischlin, Martin Franz, Stefan Katzenbeisser, Mark Manulis, Cristina Onete, Andreas Peter, Bertram Poettering, and Dominique Schröder. Redactable signatures for tree-structured data: Definitions and constructions. In *Applied Cryptography and Network Security, 8th International Conference, ACNS 2010, Beijing, China, June 22-25, 2010. Proceedings*, pages 87–104, 2010.

[BBFR15] Michael Backes, Manuel Barbosa, Dario Fiore, and Raphael M. Reischuk. ADSNARK: nearly practical and privacy-preserving proofs on authenti-

cated data. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 271–286, 2015.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 326–349, 2012.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 111–120, 2013.

[BCF+14] Julien Bringer, Hervé Chabanne, Mélanie Favre, Alain Patey, Thomas Schneider, and Michael Zohner. GSHADE: faster privacy-preserving distance computation and biometric identification. In *ACM Information Hiding and Multimedia Security Workshop, IH&MMSec'14, Salzburg, Austria, June 11-13, 2014*, pages 187–198, 2014.

[BCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. Snarks for C: verifying program executions succinctly and in zero knowledge. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 90–108, 2013.

[BCG+14] 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, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 459–474, 2014.

[BCG+15] Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. Secure sampling of public parameters for succinct zero knowledge proofs. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 287–304, 2015.

[BCI+07] Julien Bringer, Hervé Chabanne, Malika Izabachène, David Pointcheval, Qiang Tang, and Sébastien Zimmer. An application of the goldwasser-micali cryptosystem to biometric authentication. In *Information Security and Privacy, 12th Australasian Conference, ACISP 2007, Townsville, Australia, July 2-4, 2007, Proceedings*, pages 96–106, 2007.

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, pages 315–333, 2013.

[BCP13] Julien Bringer, Hervé Chabanne, and Alain Patey. SHADE: secure hamming distance computation from oblivious transfer. In *Financial Cryptography and Data Security - FC 2013 Workshops, USEC and WAHC 2013,*

Okinawa, Japan, April 1, 2013, Revised Selected Papers, pages 164–176, 2013.

[BCTV14a] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Scalable zero knowledge via cycles of elliptic curves. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, pages 276–294, 2014.

[BCTV14b] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 781–796, 2014.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 21–31, 1991.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 103–112, 1988.

[BFR13a] Michael Backes, Dario Fiore, and Raphael M. Reischuk. Verifiable delegation of computation on outsourced data. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 863–874, 2013.

[BFR+13b] Benjamin Braun, Ariel J. Feldman, Zuocheng Ren, Srinath T. V. Setty, Andrew J. Blumberg, and Michael Walfish. Verifying computations with state. In *ACM SIGOPS 24th Symposium on Operating Systems Principles, SOSP'13, Farmington, PA, USA, November 3-6, 2013*, pages 341–357, 2013.

[BFS16] Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. Nizks with an untrusted CRS: security in the face of parameter subversion. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*, pages 777–804, 2016.

[BGG+88] Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. In *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, pages 37–56, 1988.

[BGM17] Sean Bowe, Ariel Gabizon, and Ian Miers. Scalable multi-party computation for zk-snark parameters in the random beacon model. *IACR Cryptology ePrint Archive*, 2017:1050, 2017.

[BGV11] Siavosh Benabbas, Rosario Gennaro, and Yevgeniy Vahlis. Verifiable delegation of computation over large datasets. In *Advances in Cryptology -*

CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings, pages 111–131, 2011.

[Blu81] Manuel Blum. Coin flipping by telephone. In *Advances in Cryptology: A Report on CRYPTO 81, CRYPTO 81, IEEE Workshop on Communications Security, Santa Barbara, California, USA, August 24-26, 1981.*, pages 11–15, 1981.

[Bou00] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, pages 431–444, 2000.

[BR05] Mihir Bellare and Phillip Rogaway. Introduction to modern cryptography. In *UCSD CSE 207 Course Notes*, page 207, 2005.

[BS] Dan Boneh and Victor Shoup. A graduate course in applied cryptography. Version 0.3. `http://cryptobook.us` Accessed: 2017-07-25.

[BSSC05] I. Blake, G. Seroussi, N. Smart, and J. W. S. Cassels. *Advances in Elliptic Curve Cryptography*. London Mathematical Society Lecture Note Series. Cambridge University Press, 2005.

[CFH+15] Craig Costello, Cédric Fournet, Jon Howell, Markulf Kohlweiss, Benjamin Kreuter, Michael Naehrig, Bryan Parno, and Samee Zahur. Geppetto: Versatile verifiable computation. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 253–270, 2015.

[CFPZ09] Céline Chevalier, Pierre-Alain Fouque, David Pointcheval, and Sébastien Zimmer. Optimal randomness extraction from a diffie-hellman element. In *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, pages 572–589, 2009.

[CGGN17] Matteo Campanelli, Rosario Gennaro, Steven Goldfeder, and Luca Nizzardo. Zero-knowledge contingent payments revisited: Attacks and payments for services. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 229–243, 2017.

[CHK17] Hervé Chabanne, Rodolphe Hugel, and Julien Keuffer. Verifiable document redacting. In *Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part I*, pages 334–351, 2017.

[CMT12] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 90–112, 2012.

[CP92] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, pages 89–105, 1992.

[CRR11] Ran Canetti, Ben Riva, and Guy N. Rothblum. Practical delegation of computation using multiple servers. In *Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, Chicago, Illinois, USA, October 17-21, 2011*, pages 445–454, 2011.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, pages 13–25, 1998.

[CT10] Alessandro Chiesa and Eran Tromer. Proof-carrying data and hearsay arguments from signature cards. In *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*, pages 310–331, 2010.

[CTY11] Graham Cormode, Justin Thaler, and Ke Yi. Verifying computations with streaming interactive proofs. *PVLDB*, 5(1):25–36, 2011.

[DPSS15] David Derler, Henrich C. Pöhls, Kai Samelin, and Daniel Slamanig. A general framework for redactable signatures and new constructions. In *Information Security and Cryptology - ICISC 2015 - 18th International Conference, Seoul, South Korea, November 25-27, 2015, Revised Selected Papers*, pages 3–19, 2015.

[FG12] Dario Fiore and Rosario Gennaro. Publicly verifiable delegation of large polynomials and matrix computations, with applications. In *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 501–512, 2012.

[Fis01] Marc Fischlin. *Trapdoor commitment schemes and their applications*. PhD thesis, Goethe University Frankfurt, Frankfurt am Main, Germany, 2001.

[For89] Lance Fortnow. The complexity of perfect zero-knowledge. *Advances in Computing Research*, 5:327–343, 1989.

[Fre77] Rusins Freivalds. Probabilistic machines can use less running time. In *IFIP Congress*, pages 839–842, 1977.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, pages 186–194, 1986.

[Fuc18] Georg Fuchsbauer. Subversion-zero-knowledge snarks. In *Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part I*, pages 315–347, 2018.

[Gam84] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, pages 10–18, 1984.

[GDL+16] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin E. Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 201–210, 2016.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178, 2009.

[GGG17] Zahra Ghodsi, Tianyu Gu, and Siddharth Garg. Safetynets: Verifiable execution of deep neural networks on an untrusted cloud. *CoRR*, abs/1706.10268, 2017.

[GGH96] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Collision-free hashing from lattice problems. *Electronic Colloquium on Computational Complexity (ECCC)*, 3(42), 1996.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, pages 465–482, 2010.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 626–645, 2013.

[GKM+18] Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-snarks. *IACR Cryptology ePrint Archive*, 2018:280, 2018.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 113–122, 2008.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 291–304, 1985.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all np-statements in zero-knowledge, and a methodology of cryptographic protocol

design. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, pages 171–185, 1986.

[Gol08] Oded Goldreich. *Computational complexity - a conceptual perspective*. Cambridge University Press, 2008.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, pages 339–358, 2006.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, pages 321–340, 2010.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, pages 305–326, 2016.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 99–108, 2011.

[HMEK11] Yan Huang, Lior Malka, David Evans, and Jonathan Katz. Efficient privacy-preserving biometric identification. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6th February - 9th February 2011*, 2011.

[IKO07] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient arguments without short pcps. In *22nd Annual IEEE Conference on Computational Complexity (CCC 2007), 13-16 June 2007, San Diego, California, USA*, pages 278–291, 2007.

[JKO13] Marek Jawurek, Florian Kerschbaum, and Claudio Orlandi. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 955–966, 2013.

[JMSW02] Robert Johnson, David Molnar, Dawn Xiaodong Song, and David Wagner. Homomorphic signature schemes. In *Topics in Cryptology - CT-RSA 2002, The Cryptographer's Track at the RSA Conference, 2002, San Jose, CA, USA, February 18-22, 2002, Proceedings*, pages 244–262, 2002.

[JW99] Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. In *CCS '99, Proceedings of the 6th ACM Conference on Computer and Communications Security, Singapore, November 1-4, 1999.*, pages 28–36, 1999.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 723–732, 1992.

[KMC18] Julien Keuffer, Refik Molva, and Hervé Chabanne. Efficient proof composition for verifiable computation. In *Computer Security - 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part I*, pages 152–171, 2018.

[KPS18] Ahmed Kosba, Charalampos Papamanthou, and Elaine Shi. xJsnark: A framework for efficient verifiable computation. In *2018 IEEE Symposium on Security and Privacy (SP)*, volume 00, pages 543–560, 2018.

[KSC09] Ghassan Karame, Mario Strasser, and Srdjan Capkun. Secure remote execution of sequential computations. In *Information and Communications Security, 11th International Conference, ICICS 2009, Beijing, China, December 14-17, 2009. Proceedings*, pages 181–197, 2009.

[KTX08] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In *Advances in Cryptology - ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7-11, 2008. Proceedings*, pages 372–389, 2008.

[KZM+15] Ahmed Kosba, Zhichao Zhao, Andrew Miller, Yi Qian, Hubert Chan, Charalampos Papamanthou, Rafael Pass, abhi shelat, and Elaine Shi. C∅c∅: A framework for building composable zero-knowledge proofs. Cryptology ePrint Archive, Report 2015/1093, 2015. `http://eprint.iacr.org/2015/1093`.

[LFKN90] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 2–10, 1990.

[lib] libsnark. Available at `https://github.com/scipr-lab/libsnark`. Accessed: September 2018.

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, pages 169–189, 2012.

[Mer87] Ralph C. Merkle. A digital signature based on a conventional encryption function. In *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, pages 369–378, 1987.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.

[MNL14] Sébastien Marcel, Mark S. Nixon, and Stan Z. Li, editors. *Handbook of Biometric Anti-Spoofing - Trusted Biometrics under Spoofing Attacks*. Advances in Computer Vision and Pattern Recognition. Springer, 2014.

[MRV99] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 120–130, 1999.

[Nat15] Secure Hash Standard (SHS). Federal Information Processing Standard 180-4, 2015. National Institute of Standards and Technology.

[NT16] Assa Naveh and Eran Tromer. Photoproof: Cryptographic image authentication for any set of permissible transformations. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 255–271, 2016.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 427–437, 1990.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, pages 223–238, 1999.

[Ped91] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, pages 129–140, 1991.

[pep] The Pepper Project - toward practical verifiable computation. Available at `https://github.com/pepper-project`. Accessed: Spetember 2018.

[PF79] Nicholas Pippenger and Michael J. Fischer. Relations among complexity measures. *J. ACM*, 26(2):361–381, 1979.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 238–252, 2013.

[PM17] Elena Pagnin and Aikaterini Mitrokotsa. Privacy-preserving biometric authentication: Challenges and directions. *Security and Communication Networks*, 2017:7129505:1–7129505:9, 2017.

[PRV12] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, pages 422–439, 2012.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, 13(3):361–396, 2000.

[RCB01] Nalini K. Ratha, Jonathan H. Connell, and Ruud M. Bolle. Enhancing security and privacy in biometrics-based authentication systems. *IBM Systems Journal*, 40(3):614–634, 2001.

[SBCS12] Koen Simoens, Julien Bringer, Hervé Chabanne, and Stefaan Seys. A framework for analyzing template security and privacy in biometric authentication systems. *IEEE Trans. Information Forensics and Security*, 7(2):833–841, 2012.

[SBV+13] Srinath T. V. Setty, Benjamin Braun, Victor Vu, Andrew J. Blumberg, Bryan Parno, and Michael Walfish. Resolving the conflict between generality and plausibility in verified computation. In *Eighth Eurosys Conference 2013, EuroSys '13, Prague, Czech Republic, April 14-17, 2013*, pages 71–84, 2013.

[SBZ01] Ron Steinfeld, Laurence Bull, and Yuliang Zheng. Content extraction signatures. In *Information Security and Cryptology - ICISC 2001, 4th International Conference Seoul, Korea, December 6-7, 2001, Proceedings*, pages 285–304, 2001.

[Sch89] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 239–252, 1989.

[Sha90] Adi Shamir. IP=PSPACE. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 11–15, 1990.

[Sho] Victor Shoup. NTL: a library for doing number theory. Available at `http://shoup.net/ntl`. Accessed: September 2018.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, pages 256–266, 1997.

[Sho06] Victor Shoup. *A computational introduction to number theory and algebra.* Cambridge University Press, 2006.

[SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 815–823, 2015.

[SMBW12] Srinath T. V. Setty, Richard McPherson, Andrew J. Blumberg, and Michael Walfish. Making argument systems for outsourced computation practical (sometimes). In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*, 2012.

[SR10] Daniel Slamanig and Stefan Rass. Generalizations and extensions of redactable signatures with applications to electronic healthcare. In *Communications and Multimedia Security, 11th IFIP TC 6/TC 11 International Conference, CMS 2010, Linz, Austria, May 31 - June 2, 2010. Proceedings*, pages 201–213, 2010.

[SSW10] Ahmad-Reza Sadeghi, Thomas Schneider, and Marcel Winandy. Token-based cloud computing. In *Trust and Trustworthy Computing, Third International Conference, TRUST 2010, Berlin, Germany, June 21-23, 2010. Proceedings*, pages 417–429, 2010.

[SVP+12] Srinath T. V. Setty, Victor Vu, Nikhil Panpalia, Benjamin Braun, Andrew J. Blumberg, and Michael Walfish. Taking proof-based verified computation a few steps closer to practicality. In *Proceedings of the 21th USENIX Security Symposium, Bellevue, WA, USA, August 8-10, 2012*, pages 253–268, 2012.

[Tha13] Justin Thaler. Time-optimal interactive proofs for circuit evaluation. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 71–89, 2013.

[TYRW14] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 1701–1708, 2014.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008.*, pages 1–18, 2008.

[VSBW13] Victor Vu, Srinath T. V. Setty, Andrew J. Blumberg, and Michael Walfish. A hybrid architecture for interactive verifiable computation. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 223–237, 2013.

[WB15] Michael Walfish and Andrew J. Blumberg. Verifying computations without reexecuting them. *Commun. ACM*, 58(2):74–84, January 2015.

[WJB+17a] Riad S. Wahby, Ye Ji, Andrew J. Blumberg, Abhi Shelat, Justin Thaler, Michael Walfish, and Thomas Wies. Full accounting for verifiable outsourcing. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 2071–2086, 2017.

[WJB+17b] Riad S. Wahby, Ye Ji, Andrew J. Blumberg, Abhi Shelat, Justin Thaler, Michael Walfish, and Thomas Wies. Full accounting for verifiable outsourcing. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 2071–2086, 2017.

[WLCS18] Shuang Wu, Guoqi Li, Feng Chen, and Luping Shi. Training and inference with integers in deep neural networks. In *International Conference on Learning Representations*, 2018.

[WSR+15] Riad S. Wahby, Srinath T. V. Setty, Zuocheng Ren, Andrew J. Blumberg, and Michael Walfish. Efficient RAM and control flow in verifiable outsourced computation. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015*, 2015.

[WTS+18] R. S. Wahby, I. Tzialla, A. Shelat, J. Thaler, and M. Walfish. Doubly-efficient zksnarks without trusted setup. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 975–992, 2018.

[YSK+13] Masaya Yasuda, Takeshi Shimoyama, Jun Kogure, Kazuhiro Yokoyama, and Takeshi Koshiba. Packed homomorphic encryption based on ideal lattices and its application to biometrics. In *Security Engineering and Intelligence Informatics - CD-ARES 2013 Workshops: MoCrySEn and SeCIHD, Regensburg, Germany, September 2-6, 2013. Proceedings*, pages 55–74, 2013.

[ZGK+17] 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, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 863–880, 2017.