DISSERTATIONES
MATHEMATICAE
UNIVERSITATIS
TARTUENSIS
109

PRASTUDY MUNGKAS FAUZI

Efficient non-interactive zero-knowledge protocols in the CRS model

DISSERTATIONES MATHEMATICAE UNIVERSITATIS TARTUENSIS

109

DISSERTATIONES MATHEMATICAE UNIVERSITATIS TARTUENSIS
109

PRASTUDY MUNGKAS FAUZI

Efficient Non-interactive Zero-knowledge Protocols in the CRS Model

UNIVERSITY OF TARTU
Press

Institute of Computer Science, Faculty of Mathematics and Computer Science, University of Tartu, Estonia

Dissertation is accepted for the commencement of the degree of Doctor of Philosophy (PhD) on 19th of December, 2016 by the Council of the Institute of Computer Science, University of Tartu.

Supervisor: Ph.D Helger Lipmaa, University of Tartu, Tartu, Estonia

Opponents: Ph.D Carla Ràfols, Pompeu Fabra University, Barcelona, Spain
Ph.D Ivan Visconti, University of Salerno, Salerno, Italy

The public defense will take place on February 17, 2017 at 14.15 in Liivi 2-405.
The publication of this dissertation was financed by the Estonian Doctoral School in Information and Communication Technology.

European Union
European Social Fund

Investing in your future

Panoramix

ISSN 1024-4212
ISBN 978-9949-77-332-9 (print)
ISBN 978-9949-77-333-6 (pdf)

University of Tartu Press
www.tyk.ee

Contents

PUBLICATIONS INCLUDED IN THE THESIS

1. Fauzi, P., Lipmaa, H., Zhang, B.: Efficient Modular NIZK Arguments from Shift and Product. In: Abdalla, M., Nita-Rotaru, C., Dahab, R. (eds.) CANS 2013. LNCS, vol. 8257, pp. 92–121. Springer International Publishing, Paraty, Brazil (Nov 20–22, 2013).

2. Fauzi, P., Lipmaa, H., Zhang, B.: Efficient Non-Interactive Zero Knowledge Arguments for Set Operations. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 216–233. Springer Berlin Heidelberg, Bridgetown, Barbados (March 3–7, 2014).

3. Fauzi, P., Lipmaa, H.: Efficient Culpably Sound NIZK Shuffle Argument without Random Oracles. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 200–216. Springer International Publishing, San Franscisco, CA, USA (February 29–March 4, 2016).

4. Fauzi, P., Lipmaa, H., Zając, M.: A Shuffle Argument Secure in the Generic Model. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016 (2). LNCS, vol. 10032, pp. 841–872. Springer Berlin Heidelberg, Hanoi, Vietnam (Dec 4–8 2016).

ABSTRACT

In the current digital era, we can do increasingly astonishing activities remotely using only our electronic devices. In all these activities, cryptographic protocols are required to ensure privacy for the users. In reality, some parties participating in a protocol may be dishonest, and not act according to what was agreed in the protocol specification. Hence, for a real world protocol to be secure, we also need each party to prove that it behaves correctly, in accordance to the protocol. However, it is often difficult for a party to do so without sacrificing privacy of its inputs. One way to achieve this is by constructing a zero-knowledge argument: a proof that gives nothing else away besides the correctness of the statement, and is sound against polynomial-time provers who are dishonest.

In many cases, we want a zero-knowledge argument to be non-interactive and transferable, so that it needs to be computed only once, but can be verified by many verifiers at any future time. An interactive zero-knowledge argument can be made non-interactive (say) using the Fiat-Shamir heuristic, where a verifier's messages to the prover are replaced by a uniformly random output from the same message domain, resulting in a secure protocol in the random oracle (RO) model. Another way to achieve non-interactivity is by using the common reference string (CRS) model, where a trusted third party outputs a common string that removes the need for an honest verifier's response but still guarantees security. Due to some limitations of the RO model, we prefer working in the CRS model.

In this thesis we provide three scenarios where non-interactive zero-knowledge (NIZK) arguments in the CRS model can be made more efficient, and are comparable in efficiency to the best known NIZK argument in the RO model. First, we explain the need for verifiable computation. In this scenario we get more efficient CRS-model NIZK arguments for $\mathbf{NP}$-complete languages that are simpler to check than CIRCUIT-SAT. Next, we discuss the challenge of authorization in real world situations. In this scenario we get NIZK arguments for set operations that are as efficient than existing ones in the RO model but for a bigger library of set operations. Finally, we discuss the need to shuffle ciphertexts in electronic voting. In this scenario we get two efficient CRS-model NIZK shuffle arguments that are almost as efficient as existing ones in the RO model.

CHAPTER 1

INTRODUCTION

1.1 The Need to Verify Procedures

In the current digital era, we can do increasingly astonishing activities remotely using only our electronic devices. Using mobile applications such as WhatsApp, we can message or call someone with the guarantee, using an end-to-end encryption protocol, that only the recipient can know the contents of the message or conversation. Most banking systems enable us to pay our bills and perform other financial transactions, and use the TLS protocol to guarantee that no one can read or modify the transaction data. Some countries provide an option to vote electronically in an election (e.g. Estonia) or referendum (e.g. Switzerland) with similar privacy guarantees to traditional paper voting. In all these activities, a protocol is required to ensure privacy for the users. These protocols rely heavily in cryptography, the science of secure communication between two or more parties, and hence are categorized as cryptographic protocols.

Cryptographic protocols are much more than just ensuring privacy. This is because in reality, we cannot just assume that all parties participating in a protocol will act according to what was agreed in the protocol specification. Hence, for a real world protocol to be secure, we also need each party to prove that all procedures it performs are in accordance to the protocol. As we want the protocol to be practical, it must not take too long to construct such a proof. Moreover, in many cases, the verifier does not have the computing power of a prover. For instance, in the scenario of verifiable computation [48], where a verifier outsources some computation-intensive operation to a super-computer, the verifier can be lightweight (e.g. a mobile device). In such cases, other parties must be able to quickly verify these procedures are indeed performed as intended.

However, it is often difficult for a party to prove that it acts correctly without sacrificing privacy of its inputs. For example, consider the case of electronic voting. A voter must prove that he votes for a valid candidate from the set of candidates, but it is difficult to do so without giving away information on who he voted for. If the electronic voting protocol involves mix networks (i.e. networks that remove the relationship between senders and their messages [79]) to mix the encrypted votes before the decryption process, then these mix networks must prove that they perform the mixing correctly, but it is difficult to do so without revealing something about the permutation used while mixing. Hence, what we really want is a proof that gives nothing else away besides the correctness of the statement. In cryptography, this is known as a zero-knowledge proof [54]. In its simplest form, we can assume only two roles: a prover who creates a zero-knowledge proof, and verifiers who either accept or reject the proof.

In many cases, it is not ideal for a zero-knowledge proof to be interactive. Consider the case of an independent audit of an election based on electronic voting, performed some time after the election. The voters and voting servers involved in the process must get back online to recreate proofs and possibly redo many costly operations. In an interactive proof, a prover will have to compute proofs for each verifier, and must be online during the whole process. In contrast, a non-interactive proof [32] can be checked at any future time, without requiring the prover to be online. Moreover, if the non-interactive proof is transferable (i.e., a verifier can directly transfer an accepting proof to another verifier), the prover can compute a proof just once, and this proof can be checked by many verifiers. As we will see in subsequent chapters, transferable proofs are essential in many common scenarios. Non-interactive proofs that are non-transferable (i.e., an accepting proof cannot be transferred to another verifier) are also useful in some scenarios such as identification [21, 13], but these scenarios are not addressed in this thesis.

1.2 Scope and Claim of This Thesis

In cryptography we are familiar with two security models, passive and active. In the passive security model, all parties participating in the protocol are assumed to follow the protocol honestly, and the only mischief they can do is to be curious about what other parties' private inputs can be. In the active security model, we have no guarantees that participating parties act according to the protocol. In this case, each party must provide a guarantee of correct execution.

As we discussed above, we will focus on the active security model for two-party protocols, where correct execution is guaranteed by a zero-knowledge proof. Moreover, one party is designated as a prover who must prove some statement, while the other party is designated as a verifier who can check this proof. We focus on non-interactive proofs, consisting of a single message from the prover to the verifier.

There are two models mainly used to get non-interactive zero-knowledge (NIZK) proofs: the random oracle (RO) model, and the common reference string (CRS) model.

- In the RO model [50], we assume the existence of one or more random oracles that give uniformly random responses. Using the Fiat-Shamir heuristic [45] or recent transforms by Lindell [66] and Ciampi et al. [29], these random responses can be used to replace every message a verifier sends to the prover. Although protocols in the RO model are very efficient, they have some limitations. Due to the required properties, there is no finite algorithm that can fully implement a random oracle. Moreover, there exist cases where a protocol is secure in the RO model, but insecure if the RO is replaced by any hash function [25, 52, 7, 12, 19].

- In the CRS model, we assume the existence of a trusted third party that outputs a common string that incorporates an honest verifier's response in an interactive protocol. After producing this string, the trusted third party will not be needed anymore. A prover can reference this string while creating a proof, and a verifier can use it during verification. The main disadvantage of the CRS model is that it requires trust that the third party computes the CRS correctly. However, in contrast to the RO model, this eliminates the need for a heuristic and how to securely instantiate the RO. Hence security proofs in the CRS model are more convincing than the Fiat-Shamir heuristic used in the RO model.

We note that the Lindell [66] and Ciampi et al. [29] transforms have improved heuristic security compared to the Fiat-Shamir heuristic, but they use both RO and CRS. Additionally, there are two models which can be seen as relaxations of the CRS model: the registered public key (RPK) model [4], and the bare public key (BPK) model [24]. However, in both cases, the non-interactive proof is only valid and checkable by a single verifier, and hence not transferable. While this in itself is not a bad thing, it is outside our scope of transferable NIZK arguments.

In this thesis, we will focus solely on NIZK protocols in the CRS model. This is because we are interested in cases where the proofs need to be computed just once, and hence need to be non-interactive and transferable. Moreover, we want to provide an alternative to existing NIZK protocols in the RO model. We make a further restriction that we assume even a malicious prover runs in polynomial time, in which case we get NIZK proofs that are computationally sound, instead of perfectly sound. This is because it is widely believed that perfectly sound proofs have some efficiency limitations. For example, Killian [64] proved that while computationally sound zero-knowledge proofs can be succinct, it is unlikely to be the case for perfectly sound zero-knowledge proofs. Computationally sound proofs are commonly called arguments, and similarly computationally sound NIZK proofs are commonly called NIZK arguments.

The main result of this thesis is that in many practical scenarios, secure NIZK arguments in the CRS model can be made as efficient as those in the RO model. To support this claim, we will provide three interesting scenarios where we construct NIZK arguments in the CRS model with efficiency comparable to the best known NIZK argument in the RO model. In the first scenario, a lightweight verifier outsources computation to a much more powerful prover, and needs to verify that the computation was indeed done correctly. In the second scenario, a lightweight verifier tries to check if a prover's private data satisfies some properties, which can be written as a set relation. In the third scenario, a prover tries to ensure the privacy of voters in electronic voting by shuffling the input ciphertexts before it goes to the decryption process, and must prove it does so correctly to a verifier with as much computational power as the prover. The resulting NIZK arguments in the third scenario are not yet as efficient as those in the RO model, but are still more efficient than existing ones in the CRS model.

In all these scenarios, the prover must provide a proof to the verifier without revealing anything about its private data (e.g., side information related to a computation, sets that contain private information, or permutation used to perform a shuffle) by using one or more NIZK arguments. This is done by first committing [28] to a set of values, then proving that the committed values satisfy some equations that cannot hold for a dishonest prover. These scenarios and the resulting NIZK arguments will be discussed in more detail in chapters 3-5.

1.3 Thesis Outline and Author's Contributions

In the following, we outline the contents of each chapter in the thesis, and describe the author's main contributions towards the co-authored papers.

Chapter 2 provides a quick overview of the basic concepts that are used in cryptographic protocols. In particular, in this chapter we will introduce the polynomial commitment scheme, which we use throughout this thesis to commit to a vector of integers, and its security properties. We also mention the various assumptions used in the subsequent chapters.

Chapter 3 introduces the need for verifiable computation, and why it is relevant to cryptographic protocols. In this case, a verifier outsources computation (e.g., solving a $\mathbf{NP}$-complete problem) to a prover who has much more computing power than the verifier. Hence an important requirement is to have very little communication and verifier's computation.

The chapter refers to the following paper included in this thesis.

- Fauzi, P., Lipmaa, H., Zhang, B.: Efficient Modular NIZK Arguments from Shift and Product. In: Abdalla, M., Nita-Rotaru, C., Dahab, R. (eds.) CANS 2013. LNCS, vol. 8257, pp. 92–121. Springer International Publishing, Paraty, Brazil (Nov 20–22, 2013).

  This paper explains two basic NIZK arguments in the CRS model, i.e. product and shift, and how these can be used to build a NIZK argument for any language in $\mathbf{NP}$, including an efficient range argument. The main improvement from previous papers is the use of best results in progression-free sets to improve existing product arguments, and the use of a shift argument instead of the more costly permutation argument used in previous work [67, 26] to get a NIZK argument for various $\mathbf{NP}$-complete languages. The author's main contribution is in constructing NIZK arguments for simple $\mathbf{NP}$-complete languages along with their security proof.

Chapter 4 takes a closer look at the challenge of authorization in real world situations. We note that many such situations involve proving some set relation, such as set membership and set intersection. Moreover, to ensure privacy we need to hide one or more of the sets in the set relation.

The chapter refers to the following paper included in this thesis.

- Fauzi, P., Lipmaa, H., Zhang, B.: Efficient Non-Interactive Zero Knowledge Arguments for Set Operations. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 216–233. Springer Berlin Heidelberg, Bridgetown, Barbados (March 3–7, 2014).

  This paper proposes a new NIZK argument to prove a simple relation, PMSET, between four committed multisets, and how these can be used to construct set membership and range arguments, as well as other set relations. The main novelty of this paper is the construction of the PMSET argument that is as efficient as existing NIZK arguments for set relations in the RO model. Moreover, the flexibility of PMSET enables us to construct many more set relations than previous work, using a constant number of PMSET arguments. The author's contributions include a detailed comparison between our arguments with related ones, constructing secure NIZK arguments for various set operations from PMSET, and parts of the final security proof.

Chapter 5 introduces the concept of shuffling ciphertexts and why it is useful in the case of electronic voting. In particular, if the ciphertexts are correctly shuffled before being sent to the decryption entity, the relationship between voters and their votes can be completely removed. The shuffling is done sequentially by a chain of mix-servers, all of which need to efficiently prove that it shuffles correctly, and efficiently verify that the previous mix-servers computed their shuffles correctly.

This chapter contains the most interesting scenario compared to previous chapters, as it contains the bulk of the author's research work, culminating in the following two papers included in this thesis.

- Fauzi, P., Lipmaa, H.: Efficient Culpably Sound NIZK Shuffle Argument without Random Oracles. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 200–216. Springer International Publishing, San Franscisco, CA, USA (February 29–March 4, 2016).

  This paper proposes a new and more efficient shuffle argument in the CRS model, based on a new computational assumption. The main technical novelty of this paper is the use of this new computational assumption in addition to recent results in square-span programs [30] to create a culpably sound shuffle argument. The author constructed a sub-argument and the full shuffle argument and its security proof, which was later perfected by his supervisor (and co-author). The author also improved the security proof of a sub-argument and the new computational assumption, which led to an optimized shuffle argument.

- Fauzi, P., Lipmaa, H., Zając, M.: A Shuffle Argument Secure in the Generic Model. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016 (2). LNCS, vol. 10032, pp. 841–872. Springer Berlin Heidelberg, Hanoi, Vietnam (Dec 4–8 2016).

  This paper proposes an even more efficient shuffle argument, where the biggest improvement is in verifier's complexity, which is the bottleneck for electronic voting protocols based on mix networks. Another notable difference from the previous paper is that we achieve soundness (instead of culpable soundness) without knowledge assumptions, and we get soundness proof in the generic bilinear group model. Moreover, the soundness proof of the full shuffle argument can be automated. The author's main contributions are related to the use of automated tools to aid in modifying and checking the security of sub-arguments, the use of batching to improve verification efficiency, and proving the resulting argument stays sound and zero-knowledge.

Chapter 6 summarizes the author's work related to NIZK arguments in the CRS model. It contains a summary of the specific scenarios and the contributions of the author's work in these scenarios. The chapter also shows where the author's work can be improved even further.

CHAPTER 2

PRELIMINARIES

In this chapter, we define the common notation and important definitions used throughout this thesis. Note that some notation and definitions used in this thesis are slightly different compared to that used in the original publications, but are still equivalent. This is done to unify any difference in notation between the original publications.

2.1 Basic Notation

We define $\mathbb{Z}$ to be the set of integers, and $\mathbb{N}$ be the set of positive integers. For integers $A \leq B$, define $[A \mathinner{..} B] = \{A, A+1, \cdots, B\}$. For sets G, H, let $G \times H = \{(g,h) | g \in G, h \in H\}$. For a finite set S, let $|S|$ be the size of the set. For $x \in \mathbb{N}$, let $\|x\| = \lfloor \log_2(x) \rfloor + 1$ be the length of x in bits.

Definition 1. A function $f : \mathbb{N} \to \mathbb{Z}$ is *negligible* if it decreases faster than $1/P(\kappa)$ for any polynomial P. That is, f is negligible if for any polynomial P, there exists a constant $N_0 \in \mathbb{N}$ such that for all $\kappa > N_0$,

$$|f(\kappa)| < \frac{1}{P(\kappa)} \ .$$

Similarly, a function f is *noticeable* if it grows faster than $1/P(\kappa)$ for some polynomial P. Additionally, a function f is *non-negligible* if it is not a negligible function, and it is *overwhelming* if $1 - f$ is negligible.

An example of a negligible function is $f(\kappa) = 2^{-\kappa}$. An example of a noticeable function is $f(\kappa) = 2/\kappa$. Note that a non-negligible function need not be noticeable. For example, $f(\kappa) = \frac{1+(-1)^\kappa}{2}$ is neither negligible nor noticeable. We denote $\mathbf{poly}(\kappa)$ to mean an unspecified function polynomial in κ, and $\mathbf{negl}(\kappa)$ to mean an unspecified function negligible in κ.

2.2 Groups

Definition 2. Given a set G and operation $\circ$, $(G, \circ)$ is a *group* if the following conditions hold.

1. The operation $\circ$ is associative and closed in G.
2. G contains an identity element (denoted as 1_G) such that for all $g \in G$
$$1_G \circ g = g \circ 1_G = g \ .$$
3. Every $g \in G$ has an inverse in G (denoted by g^{-1}) such that
$$g^{-1} \circ g = g \circ g^{-1} = 1_G \ .$$

Moreover, if $\circ$ is commutative then we have an *Abelian* group. If the operation is obvious from the context, we will write the group as just G. An element $g \in G$ is a *generator* of G if for all $h \in G$ there exists an integer i such that $h = g^i$. If a group has a generator, then it is *cyclic*, in which case it is also abelian. In this work, we primarily use groups of prime order, which are always cyclic.

Let G, H be groups of order p. Then the direct product $G \times H$ with the operation component-wise multiplication is also a group. The product and exponentiation operations are defined as follows.

- For $(g, h), (g', h') \in G \times H$: $(g, h) \cdot (g', h') = (g \cdot g', h \cdot h')$.
- For $(g, h) \in G \times H$ and $a \in \mathbb{Z}_p$: $(g, h)^a = (g^a, h^a)$.

For a group G, define G^* to be the set of non-identity elements in G. For a group G, a group element $g \in G$ and a finite set of integers $S = (s_1, \cdots, s_n)$, define $g^S = (g^{s_1}, \cdots, g^{s_n})$.

2.3 Bilinear Maps

Let A, B, and C be additive groups over a field F. A map $f : A \times B \to C$ is *bilinear* if it is linear in both arguments. That is, the following conditions hold.

- For all $a_1, a_2 \in A, b \in B$: $f(a_1 + a_2, b) = f(a_1, b) + f(a_2, b)$.
- For all $a \in A, b_1, b_2 \in B$: $f(a, b_1 + b_2) = f(a, b_1) + f(a, b_2)$.
- For all $a \in A, b \in B$ and scalar $c \in F$: $f(c \cdot a, b) = c \cdot f(a, b) = f(a, c \cdot b)$.

In cryptography, we define bilinear maps with added properties, and often call it a *pairing* [5]. It is also common to use multiplicative notation.

Definition 3. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be cyclic multiplicative groups of prime order p. A function $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a *bilinear map* (or *pairing*) if the following properties hold.

- For all $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2, a, b, \in \mathbb{Z}_p$: $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$.
- The map $\hat{e}$ is efficiently computable.
- The map $\hat{e}$ is non-degenerate, i.e., if $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ are not identity elements, then neither is $\hat{e}(g_1, g_2) \in \mathbb{G}_T$.

2.4 Additive Combinatorics

The following notation is from Tao and Vu [83]. For sets X_1, X_2 and integer k, define the following operations.

Table 2.1: Operations in additive combinatorics

Operation name	Definition
Sum	$X_1 + X_2 = \{x_1 + x_2 \mid x_1 \in X_1, x_2 \in X_2\}$
Difference	$X_1 - X_2 = \{x_1 - x_2 \mid x_1 \in X_1, x_2 \in X_2\}$
Iterated sumset	For $k > 0$, $kX = \{x_1 + x_2 + \cdots + x_k \mid x_i \in X\}$
Dilation	$k \cdot X = \{kx \mid x \in X\}$
Restricted sumset	$2\widehat{\ }X = \{x_1 + x_2 \mid x_1, x_2 \in X, x_1 \neq x_2\}$

Definition 4. A set S of integers is *progression-free* if no three distinct elements create an arithmetic progression.

Note that if $x_i, x_j, x_k \in S$ with $x_i < x_j < x_k$ are an arithmetic progression, then $x_j - x_i = x_k - x_j \iff x_i + x_k = 2x_j$. If S is progression-free, then this only happens when $i = j = k$. Using additive notation, a progression-free S must satisfy $2\widehat{\ }S \cap 2 \cdot S = \emptyset$. Since $2 \cdot S \cup 2\widehat{\ }S = 2S$, then S is progression-free iff $2S$ can be partitioned into $2\widehat{\ }S$ and $2 \cdot S$.

For a positive integer N, let $r_3(N)$ be the cardinality of the largest progression-free set $S \subseteq [1 \mathinner{..} N]$. The best known lower bound for $r_3(N)$ is given by Elkin [36].

Theorem 1. *Let $N \in \mathbb{N}$, and let $r_3(N)$ be the cardinality of the largest progression-free set $S \subseteq [1 \mathinner{..} N]$. Then*

$$r_3(N) = \Omega\left(\frac{N \cdot \log^{1/4} N}{2^{2\sqrt{2\log_2 N}}}\right) .$$

From Theorem 1, we can also get a rough upper bound for $r_3^{-1}(N)$, the smallest value s_N such that the set $[1 \mathinner{..} s_N]$ contains a progression-free set of size N.

Corollary 1. *Let $N \in \mathbb{N}$, and let $r_3^{-1}(N)$ be the smallest value s_N such that the set $[1 \mathinner{..} s_N]$ contains a progression-free set of size N. Then*

$$r_3^{-1}(N) = N^{1+o(1)} \ .$$

Proof. Let $k = r_3^{-1}(N)$. Then by Theorem 1 and using the fact that r_3 is increasing, we get that

$$N = r_3(k) = \Omega\left(\frac{k \cdot \log^{1/4} k}{2^{2\sqrt{2\log_2 k}}}\right) \ .$$

But $\dfrac{k \cdot \log^{1/4} k}{2^{2\sqrt{2\log_2 k}}} = k^{1-o(1)}$. Moreover, by definition $N \leq k$. So we get that $N = k^{1-o(1)}$, which implies $k = N^{1+o(1)}$. $\square$

2.5 Multisets

Definition 5. A *multiset* is an object which is similar to a set, but with the added property that an element can appear multiple times.

For example, $\mathcal{A} = \{a, a, b, b, c\}$ and $\mathcal{B} = \{a, a, b, c, c, c\}$ are different multisets, but are equivalent as sets. Here, the difference is that the element b has multiplicity 2 in $\mathcal{A}$, but has multiplicity 1 in $\mathcal{B}$, and the element c has multiplicity 1 in $\mathcal{A}$, but has multiplicity 3 in $\mathcal{B}$.

Multiset operations are analogous to the corresponding set operations, but with the difference that the multiplicity of each element is taken into account. For multiset union we take the maximum multiplicity for each element, while for multiset intersection, we take the minimum multiplicity for each element. Moreover, we have an added operation $\uplus$ called *multiset sum*, where the multiplicity of each element is added. (Further details can be seen in [82].) Using the previous example, we can see that $\mathcal{A} \cup \mathcal{B} = \{a, a, b, b, c, c, c\}$, $\mathcal{A} \cap \mathcal{B} = \{a, a, b, c\}$, and $\mathcal{A} \uplus \mathcal{B} = \{a, a, a, a, b, b, b, c, c, c, c\}$.

2.6 Linear Algebra

Let n be the dimension of vectors. Define $\mathbf{0}_n$ to be the vector of all zeros, and $\mathbf{1}_n$ to be the vector of all ones. For two vectors $\boldsymbol{a}, \boldsymbol{b}$ of length n, define the Hadamard product $\boldsymbol{a} \circ \boldsymbol{b}$ to be the vector $\boldsymbol{c}$ such that for all $i \in [1 \mathinner{..} n]$, $c_i = a_i b_i$.

For a matrix $\boldsymbol{M}$, we denote the transpose of $\boldsymbol{M}$ by $\boldsymbol{M}^T$. For the rest of this discussion, we assume that vector and matrix elements are in the field $\mathbb{Z}_p$ for a prime p.

A *unit vector* $\boldsymbol{e}$ is a vector that has 1 one and $n-1$ zeros. A 1-*sparse* vector is a vector that has at most one non-zero element, i.e. it is of the form $k\boldsymbol{e}$ for some constant k and unit vector $\boldsymbol{e}$. The following lemma that characterizes unit vectors is a folklore result.

Lemma 1. *A vector $\boldsymbol{a} \in \mathbb{Z}_p^n$ is a unit vector iff the following two conditions hold.*

1. $\boldsymbol{a} \circ \boldsymbol{a} = \boldsymbol{a}$.
2. $\displaystyle\sum_{i=1}^{n} a_i = 1$.

Proof. ($\Longrightarrow$) If $\boldsymbol{a}$ is a unit vector, then it obviously satisfies the two conditions. ($\Longleftarrow$) Assume both conditions hold for a vector $\boldsymbol{a}$. Then from the first condition, we have that $a_i^2 = a_i$, so $a_i \in \{0, 1\}$. Hence for some $k \in [1\,..\,n]$, $\boldsymbol{a}$ contains k one and $n-k$ zeros. But from the second condition, $1 = \displaystyle\sum_{i=1}^{n} a_i = k$. Hence $\boldsymbol{a}$ is a unit vector. $\square$

A *permutation matrix* is a square matrix that is obtained by permuting the rows of the identity matrix I_n, based on some permutation ψ. The following lemma (and its corollary) that characterizes permutation matrices is also a folklore result, a proof of which can be found in [71].

Lemma 2. *A square matrix $\boldsymbol{M} \in \mathbb{Z}_q^{n\times n}$ with rows $(\boldsymbol{M}_i)_{i=1}^n$ is a permutation matrix iff the following two conditions hold.*

1. *For i $\in [1\,..\,n]$, $\boldsymbol{M}_i^T$ is a 1-sparse vector.*
2. *Every column of $\boldsymbol{M}$ sums to 1. (Equivalently, $\displaystyle\sum_{i=1}^{n} \boldsymbol{M}_i = \boldsymbol{1}_n$.)*

Corollary 2. *A square matrix $\boldsymbol{M} \in \mathbb{Z}_q^{n\times n}$ with rows $(\boldsymbol{M}_i)_{i=1}^n$ is a permutation matrix iff the following two conditions hold.*

1. *For i $\in [1\,..\,n]$, $\boldsymbol{M}_i^T$ is a unit vector.*
2. *Every column of $\boldsymbol{M}$ sums to 1. (Equivalently, $\displaystyle\sum_{i=1}^{n} \boldsymbol{M}_i = \boldsymbol{1}_n$.)*

2.7 Computational Assumptions

Mathematical proofs use a technique called reduction, where the fact to be proven is reduced to a previously known fact. In complexity theory and cryptography, proving a certain problem is hard is done by a series of mathematically sound steps to reduce this problem to a previously known problem which is assumed to be hard. If the reduction is done in polynomial time, then these two problems are said to be polynomial-time equivalent.

In complexity theory, most problems are measured by hardness in the *worst case*. However, in cryptography, we are most interested in hardness of computational problems in the *average case*. Moreover, security is defined for some particular input of some fixed size. This introduces non-uniformity, as an adversary may be given this input as an advice string to help solve computational problems of the same size. For a cryptographic scheme to be secure against a non-uniform probabilistic polynomial time (NUPPT) adversary, breaking the scheme must be equivalent to solving a problem which is hard in the average case. Hardness in the average case means that a NUPPT adversary has negligible probability of solving the problem.

In some complexity classes, e.g. EXP and PSPACE, worst-case hardness implies average-case hardness [77]. However, there is no known worst-case to average-case reduction for $\mathbf{NP}$-complete problems. Hence in cryptography we use problems that are assumed to be hard in the average case instead of concrete $\mathbf{NP}$-complete problems.

We will now list the computational assumptions that are used in this work. We always assume a NUPPT adversary $\mathcal{A}$, who is successful if he can output a solution to the computational problem with probability non-negligible in the security parameter κ.

The following assumptions are related to a single group generator Gen which is assumed to output a group description $\mathsf{gk} = (p, \mathbb{G})$. For example, if the X assumption holds for the chosen Gen, then we say that Gen is X-secure. We assume that the adversary also gets a generator g of $\mathbb{G}$. These two assumptions are classical and have been well-studied.

1. DL (Discrete Log, [73]) assumption: given values (g, g^{χ}) for a random $\chi \in \mathbb{Z}_p$, $\mathcal{A}$ has negligible probability of producing χ.

$$\Pr[\mathsf{gk} \leftarrow \mathsf{Gen}(1^{\kappa}), \chi \leftarrow_r \mathbb{Z}_p, \mathcal{A}(\mathsf{gk}; (g, g^{\chi})) = \chi] = \mathbf{negl}(\kappa) \ .$$

2. DDH (Decisional Diffie-Hellman, [14]) assumption: given values (g, g^a, g^b, g^c), where $c = (1-x)ab + xr$ for a random $x \in \{0, 1\}$ and random $a, b, r \in \mathbb{Z}_p$, $\mathcal{A}$ has $1/2 \pm \mathbf{negl}(\kappa)$ probability of successfully guessing

whether $c = ab$ or c is random.

$$\left| \Pr \begin{bmatrix} \mathsf{gk} \leftarrow \mathsf{Gen}(1^\kappa), a, b, r \leftarrow_r \mathbb{Z}_p, x \leftarrow_r \{0,1\}, \\ c \leftarrow (1-x)ab + xr, \mathcal{A}(\mathsf{gk}; (g, g^a, g^b, g^c)) = x \end{bmatrix} - 1/2 \right| = \mathbf{negl}(\kappa) \ .$$

The following assumptions are related to a chosen bilinear map generator BP which is assumed to output a group description $\mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$. For example, if the X assumption holds for the chosen BP, then we say that BP is X-secure. We assume that BP depends on both the security parameter κ and a value $n = \mathbf{poly}(\kappa)$ which we call the input length. We also assume that $d(n)$ and $d^*(n)$ are two functions such that $1 < d(n) < d^*(n) = \mathbf{poly}(\kappa)$, χ is generated randomly. Let $\boldsymbol{\phi} = (\phi_i)_{i=0}^n$ be a tuple of linearly independent polynomials of degree at most $d(n)$. Let g_1 be a generator of $\mathbb{G}_1$ and let g_2 be a generator of $\mathbb{G}_2$. The XDH and 2-ILin assumptions are standard and have been well-studied. The PSDL, PCDH and TSDH assumptions are less known but still well-established in the literature of non-interactive zero-knowledge in the common reference string model.

1. XDH (eXternal Diffie-Hellman, [2]) assumption in $\mathbb{G}_1$: the DDH assumption holds in $\mathbb{G}_1$, even if the adversary can access all groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ and the bilinear map $\hat{e}$.

$$\left| \Pr \begin{bmatrix} \mathsf{gk} \leftarrow \mathsf{BP}(1^\kappa, n), a, b, r \leftarrow_r \mathbb{Z}_p, \\ x \leftarrow_r \{0,1\}, c \leftarrow (1-x)ab + xr, \\ \mathcal{A}(\mathsf{gk}; (g_1, g_1^a, g_1^b, g_1^c)) = x \end{bmatrix} - 1/2 \right| = \mathbf{negl}(\kappa) \ .$$

2. 2-ILin (2-Incremental Linear, [38]) assumption in $\mathbb{G}_1$. For a matrix $\boldsymbol{A} = (a_{ij}) \in \mathbb{Z}_p^{m \times n}$, define

$$[\boldsymbol{A}]_1 = \begin{pmatrix} g_1^{a_{1,1}} & \cdots & g_1^{a_{1,n}} \\ \vdots & \ddots & \vdots \\ g_1^{a_{m,1}} & \cdots & g_1^{a_{m,n}} \end{pmatrix} \in \mathbb{G}_1^{m \times n} \ ,$$

and define $\mathcal{D}$ to be the matrix distribution

$$\boldsymbol{D} = \begin{pmatrix} a & 0 \\ 0 & a+1 \\ 1 & 1 \end{pmatrix} \in \mathbb{Z}_p^{3 \times 2}, a \leftarrow_r \mathbb{Z}_p \ .$$

Given $\boldsymbol{A}$ taken from the distribution $\mathcal{D}$, $\mathcal{A}$ cannot distinguish $[\boldsymbol{A}\boldsymbol{w}]_1$ for a random vector $\boldsymbol{w}$ and $[\boldsymbol{u}]_1$ for a random vector $\boldsymbol{u}$. In other words,

$$||\varepsilon_1 - \varepsilon_0| - 1/2| = \mathbf{negl}(\kappa) \ ,$$

where

$$\varepsilon_0 = \Pr\left[\mathsf{gk} \leftarrow \mathsf{BP}(1^\kappa, n), \boldsymbol{A} \leftarrow D, \boldsymbol{w} \leftarrow \mathbb{Z}_p^2, \mathcal{A}(\mathsf{gk}; [\boldsymbol{A}]_1, [\boldsymbol{A}\boldsymbol{w}]_1) = 1\right] \ ,$$

$$\varepsilon_1 = \Pr\left[\mathsf{gk} \leftarrow \mathsf{BP}(1^\kappa, n), \boldsymbol{A} \leftarrow D, \boldsymbol{u} \leftarrow \mathbb{Z}_p^3, \mathcal{A}(\mathsf{gk}; [\boldsymbol{A}]_1, [\boldsymbol{u}]_1) = 1\right] \ .$$

3. $\boldsymbol{\phi}$-PSDL (Power Symmetric Discrete Logarithm, [67]) assumption:

$$\Pr\begin{bmatrix} \mathsf{gk} \leftarrow \mathsf{BP}(1^\kappa, n), \chi \leftarrow_r \mathbb{Z}_p : \\ \mathcal{A}(\mathsf{gk}; (g_1^{\varphi(\chi)}, g_2^{\varphi(\chi)})_{\varphi \in \Phi}) = \chi \end{bmatrix} = \mathbf{negl}(\kappa) \ .$$

   If $\boldsymbol{\phi} = (X^j)_{j=0}^{d(n)}$, this is also known as the $d(n)$-PSDL assumption.

4. $(d(n), d^*(n))$-PCDH (Power Computational Diffie-Hellman, [56, 49]) assumption:

$$\Pr\begin{bmatrix} \mathsf{gk} \leftarrow \mathsf{BP}(1^\kappa, n), \chi \leftarrow_r \mathbb{Z}_p : \\ \mathcal{A}(\mathsf{gk}; ((g_1, g_2)^{\chi^i})_{i \in [0, d^*(n)] \setminus \{d(n)+1\}}) = g_1^{\chi^{d(n)+1}} \end{bmatrix} = \mathbf{negl}(\kappa) \ .$$

5. $d(n)$-TSDH (Target Strong Diffie-Hellman, [16, 76]) assumption:

$$\Pr\begin{bmatrix} \mathsf{gk} \leftarrow \mathsf{BP}(1^\kappa, n), \chi \leftarrow_r \mathbb{Z}_p : \\ \mathcal{A}\left(\mathsf{gk}; ((g_1, g_2)^{\chi^i})_{i=0}^{d(n)}\right) = \left(r, \hat{e}(g_1, g_2)^{1/(\chi - r)}\right) \\ \wedge r \neq \chi \end{bmatrix} = \mathbf{negl}(\kappa) \ .$$

2.8 Knowledge Assumptions

In a knowledge assumption, we assume that if an adversary $\mathcal{A}$ can produce some value (e.g. commitment) along with an accompanying value called a knowledge component, then it essentially knows how the values were computed. This is usually formalized using an algorithm $X_\mathcal{A}$ called an *extractor* that extracts the knowledge with a polynomial overhead. Related to this, we use the notation introduced by Abe and Fehr [1], where $(A, B) \leftarrow (\mathcal{A}||X_\mathcal{A})(S)$ means that if the algorithm $\mathcal{A}$ on input S outputs a tuple of values A, then the algorithm $X_\mathcal{A}$ on the same input S outputs a tuple of values B.

In this work we will use several variants of the following knowledge assumption. The definition is taken from the work of Groth [56], but generalized to match the definition in [39].

Definition 6. Let κ be the security parameter. Let m be the number of different knowledge secrets in a concrete argument. Let $\mathcal{F} = (P_i)_{i=0}^{n}$ and $\mathcal{G}_1$ be two tuples of univariate polynomials, and let $\mathcal{G}_2$ be a tuple of m-variate polynomials.

For $i \in [1 \mathinner{..} m]$, BP is $(\mathcal{F}, \mathcal{G}_1, \mathcal{G}_2, \gamma_i)$-*PKE (Power Knowledge of Exponent) secure* if for any NUPPT adversary $\mathcal{A}$ there exists a NUPPT extractor $X_{\mathcal{A}}$, such that the following probability is negligible in κ:

$$\Pr\left[\begin{array}{l} \mathsf{gk} \leftarrow \mathsf{BP}(1^{\kappa}, n), \chi \leftarrow_r \mathbb{Z}_p, \boldsymbol{\gamma} \leftarrow_r \mathbb{Z}_p^m, \\ \boldsymbol{\gamma_{-i}} = (\gamma_1, \dots, \gamma_{i-1}, \gamma_{i+1}, \cdots, \gamma_m), \mathsf{aux} \leftarrow (g_1^{\mathcal{G}_1(\chi)}, g_2^{\mathcal{G}_2(\chi, \boldsymbol{\gamma_{-i}})}), \\ (h_1, h_2; (a_i)_{i=0}^{n}) \leftarrow (\mathcal{A} \| X_{\mathcal{A}})(\mathsf{gk}; (g_1, g_2^{\gamma_i})^{\mathcal{F}(\chi)}, \mathsf{aux}) : \\ \hat{e}(h_1, g_2^{\gamma_i}) = \hat{e}(g_1, h_2) \wedge h_1 \neq g_1^{\sum_{i=0}^{n} a_i P_i(\chi)} \end{array}\right] .$$

We can regard aux as the common auxiliary input to $\mathcal{A}$ and $X_{\mathcal{A}}$ that is generated by using benign auxiliary input generation [11]. The definition implies that aux may depend on $\boldsymbol{\gamma_{-i}}$ but not on γ_i. It is important that this auxiliary input is benign, or else there can be a possibility that it is maliciously generated to enable an adversary to break the PKE assumption. In fact, Bitansky et al. [11] proved that for every choice of $(\mathcal{F}, \mathcal{G}_1, \mathcal{G}_2, \gamma_i)$, there exists a (non-benign) auxiliary input distribution and an adversary $\mathcal{A}$ such that any NUPPT extractor $X_{\mathcal{A}}$ will fail to extract the values $(a_i)_{i=0}^{n}$ as defined in the PKE assumption.

2.9 Generic Bilinear Group Model

We will now describe a cryptographic model that describes the general activities of a NUPPT adversary in the setting of bilinear groups. It is related to the generic group model definition of Maurer [72], but extended to take pairings into account. The main motivation in introducing this model is that all known attacks, given well-chosen bilinear groups, are generic in nature.

Definition 7. Assume we have a bilinear group $\mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$ generated by the group generator $\mathsf{BP}(1^{\kappa}, n)$. Consider an oracle $\mathbf{B}$ that can store values from groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ in internal state variables $\mathsf{cell}_1, \mathsf{cell}_2, \dots$, where we allow an infinite number of state variables. The initial state consists of some values $(\mathsf{cell}_1, \mathsf{cell}_2, \dots, \mathsf{cell}_{|\mathsf{inp}|})$, which are set according to some probability distribution. Each state variable cell_i has an accompanying type $\mathsf{type}_i \in \{1, 2, T, \perp\}$. For example, if $\mathsf{type}_i = 2$, then $\mathsf{cell}_i \in \mathbb{G}_2$. If $\mathsf{type}_i = \mathsf{type}_j = k$ for some integers i, j and $k \in \{1, 2, T\}$, then we can define $\mathsf{cell}_i \cdot \mathsf{cell}_j$ as multiplication in $\mathbb{G}_k$.

We assume initially $\mathsf{type}_i = \perp$ for $i > |\mathsf{inp}|$. The oracle allows computation operations on internal state variables and queries about the internal state. No other interaction with it is possible.

A computation operation consists of selecting a (say, t-ary) operation f together with $t+1$ indices $i_1, i_2, \ldots, i_{t+1}$. Assuming inputs have the correct type, $\mathbf{B}$ computes $f(\mathsf{cell}_{i_1}, \ldots, \mathsf{cell}_{i_t})$ and stores the result in $\mathsf{cell}_{i_{t+1}}$.

In the *generic bilinear group model* (GBGM) [72, 15], a NUPPT adversary $\mathcal{A}$ operates by doing a polynomial number of calls to an oracle that only allows the set of operations $\Pi = \{\cdot, \hat{e}\}$ and a set of relations $\Sigma = \{=\}$, where

- On input $(\cdot, i_1, i_2, i_3)$: if $\mathsf{type}_{i_1} = \mathsf{type}_{i_2} \neq \perp$ then set $\mathsf{cell}_{i_3} \leftarrow \mathsf{cell}_{i_1} \cdot \mathsf{cell}_{i_2}$ and $\mathsf{type}_{i_3} \leftarrow \mathsf{type}_{i_1}$.
- On input $(\hat{e}, i_1, i_2, i_3)$: if $\mathsf{type}_{i_1} = 1$ and $\mathsf{type}_{i_2} = 2$ then set $\mathsf{cell}_{i_3} \leftarrow \hat{e}(\mathsf{cell}_{i_1}, \mathsf{cell}_{i_2})$ and $\mathsf{type}_{i_3} \leftarrow T$.
- On input $(=, i_1, i_2)$: if $\mathsf{type}_{i_1} = \mathsf{type}_{i_2} \neq \perp$ and $\mathsf{cell}_{i_1} = \mathsf{cell}_{i_2}$ then return 1. Otherwise return 0.

We assume that an adversary is successful if after a polynomial number of operation queries, he makes an equality query $(=, i_1, i_2)$, $i_1 \neq i_2$, that returns 1, but cell_{i_1} and cell_{i_2} are different functions of the initial state.

Essentially, this means that in the GBGM, a NUPPT adversary can only do generic group operations to succeed. This means that if a scheme is secure in the GBGM, the only ways to break security is by exploiting some properties in the specific group instantiation.

In the GBGM, the adversary succeeds by finding $[a]_i \neq [b]_i$ such that on input $(=, [a]_i, [b]_i)$, the GBGM oracle returns 1. If an assumption holds in the GBGM, then an adversary has negligible probability of succeeding in such an equality test. This usually involves the Schwartz–Zippel lemma [81, 86], defined as follows.

Lemma 3. *(Schwartz–Zippel) Let $\mathbb{F}$ be a field, and let S be a finite subset of $\mathbb{F}$. Let $F \in \mathbb{F}[X_1, \cdots, X_n]$ be a non-zero multivariate polynomial of degree $d \geq 0$. Then*

$$\Pr[\chi_1 \leftarrow_r S, \chi_2 \leftarrow_r S, \cdots, \chi_n \leftarrow_r S : F(\chi_1, \cdots, \chi_n) = 0] \leq d/|S| \enspace .$$

Using the Schwartz–Zippel lemma, a successful equality test implies (with overwhelming probability) a successful polynomial identity test. Proving an assumption holds in the GBGM will then mean that the corresponding polynomial $F = 0$ only if $[a]_i = [b]_i$.

It can be shown that all computational and knowledge assumptions we have previously described are secure in the GBGM. For instance, Fauzi, Lipmaa and Zhang [41] showed that the PSDL assumption holds in the GBGM. The proof that the other assumptions hold follow a similar proof technique.

2.10 Public Key Cryptosystems

Definition 8. A *public key cryptosystem* [34] (or *public key encryption scheme*) is a cryptographic tool that consists of four algorithms:

- Setup: given the security parameter κ, generates the necessary group description gk. From this we can infer the message space $\mathcal{M}$, ciphertext space $\mathcal{C}$, and randomness space $\mathcal{R}$.
- Key generation algorithm Gen: given gk, generates a public key and secret key pair $(\mathsf{pk}, \mathsf{sk})$.
- Encryption algorithm Enc: given a message $m \in \mathcal{M}$, chooses a randomness value $r \in \mathcal{R}$ and outputs a *ciphertext* $c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m; r)$. For $m \in \mathcal{M}$, define $\mathsf{Enc}_{\mathsf{pk}}(m)$ to be the distribution of encryptions of m over the ciphertext space $\mathcal{C}$.
- Decryption algorithm Dec: given a ciphertext $c \in \mathcal{C}$ and secret key sk, outputs a message $m \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c)$. Decryption is deterministic, and is well defined if for any $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{Gen}(\mathsf{gk}), m \in \mathcal{M}, r \in \mathcal{R}$,

$$\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m; r)) = m \ .$$

We now introduce the most common security definition used for public key cryptosystems. There exist other security definitions, but they are not relevant to this work.

Definition 9. A probabilistic public key cryptosystem $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, is *indistinguishable under chosen plaintext attack* (IND-CPA) [53] if for any NUPPT adversary $\mathcal{A}$, there is $1/2 \pm \mathbf{negl}(\kappa)$ probability of correctly distinguishing between an encryption of two messages m_0, m_1, even if the messages were chosen by the adversary himself.

Formally, $2 \cdot |\varepsilon - 1/2| = \mathbf{negl}(\kappa)$, where

$$\varepsilon = \Pr\left[\begin{array}{l}\mathsf{gk} \leftarrow \mathsf{Setup}(1^\kappa), (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{gk}), (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pk}), \\ x \leftarrow_r \{0,1\}, r \leftarrow_r \mathcal{R}, \mathcal{A}(\mathsf{gk}; \mathsf{pk}, \mathsf{Enc}_{\mathsf{pk}}(m_x; r)) = x\end{array}\right] \ .$$

2.10.1 ElGamal Cryptosystem

A well-known public key cryptosystem that is IND-CPA secure is ElGamal [35], which is defined as follows.

- $\mathsf{Setup}(1^\kappa)$: choose a group generator GGen where the DDH assumption is assumed to hold. Set $(p, \mathbb{G}) \leftarrow \mathsf{GGen}(1^\kappa)$, and output $\mathsf{gk} \leftarrow (p, \mathbb{G})$. Here, $\mathcal{M} = \mathbb{G}$, $\mathcal{R} = \mathbb{Z}_p$, and $\mathcal{C} = \mathbb{G}^2$.

- Key generation $\mathsf{Gen}(\mathsf{gk})$: choose $g \leftarrow \mathbb{G}^*$, set $x \leftarrow_r \mathbb{Z}_p^*$, $h \leftarrow g^x$ and output $(\mathsf{pk} \leftarrow (g, h), \mathsf{sk} \leftarrow x)$.
- Encryption: given a message $m \in \mathbb{G}$ with randomizer $r \in \mathbb{Z}_p$, output $\mathsf{Enc}_{\mathsf{pk}}(m; r) \leftarrow (g^r, mh^r)$.
- Decryption: given a ciphertext $c = (c_1, c_2) \in \mathbb{G}^2$, output $\mathsf{Dec}_{\mathsf{sk}}(c) \leftarrow c_2/c_1^{\mathsf{sk}}$.

Note that decryption is well defined, since for all $m \in \mathbb{G}$, $r \in \mathbb{Z}_p$,

$$\begin{aligned} \mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m; r)) &= \mathsf{Dec}_{\mathsf{sk}}(g^r, mh^r) \\ &= mh^r/(g^r)^x \\ &= m \ . \end{aligned}$$

A variation of ElGamal known as *lifted* ElGamal is similar but with $\mathcal{M} = \mathbb{Z}_p$. The encryption algorithm is then $\mathsf{Enc}'_{\mathsf{pk}}(m; r) = \mathsf{Enc}_{\mathsf{pk}}(g^m; r)$, and the decryption algorithm is $\mathsf{Dec}'_{\mathsf{sk}}(c) = \log_g \mathsf{Dec}_{\mathsf{sk}}(c)$. Since it requires a discrete logarithm computation, the decryption algorithm is only efficient if the value of m is sufficiently small, e.g. $m < 2^4 0$.

It is well-known that ElGamal (and its lifted version) is IND-CPA secure under the DDH assumption [84]. In the bilinear group setting with group description $\mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$, we can similarly define ElGamal in group $\mathbb{G}_1$ with $\mathcal{M} = \mathbb{G}_1$, $\mathcal{R} = \mathbb{Z}_p$, and $\mathcal{C} = \mathbb{G}_1^2$. In this setting, ElGamal is IND-CPA secure under the XDH assumption in $\mathbb{G}_1$ (the proof is similar to [84], but with DDH replaced by XDH).

2.10.2 lLin Cryptosystem

A public key cryptosystem that is designed specifically for the bilinear group setting is lLin [38]. Escala et al. proved that it is IND-CPA secure under the 2-lLin assumption. The lLin cryptosystem can be IND-CPA secure in cases when ElGamal is not, and hence lLin is the better alternative in such situations. For ciphertexts in $\mathbb{G}_1$, it can be defined as follows.

- $\mathsf{Setup}(1^\kappa)$: choose a bilinear map generator BP where the 2-lLin assumption is assumed to hold. Set $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathsf{BP}(1^\kappa)$, and output $\mathsf{gk} \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$. Here, $\mathcal{M} = \mathbb{G}_1$, $\mathcal{R} = \mathbb{Z}_p^2$, and $\mathcal{C} = \mathbb{G}_1^3$.
- Key generation $\mathsf{Gen}(\mathsf{gk})$: choose $g \leftarrow \mathbb{G}_1^*$, set $x \leftarrow_r \mathbb{Z}_p \setminus \{0, -1\}$, $h \leftarrow g^x$ and output $(\mathsf{pk} \leftarrow (g, h), \mathsf{sk} \leftarrow x)$.
- Encryption: given a message $m \in \mathbb{Z}_p$ with randomizers $s_1, s_2 \in \mathbb{Z}_p$, output

$$\mathsf{Enc}_{\mathsf{pk}}(m; s_1, s_2) \leftarrow (h^{s_1}, (gh)^{s_2}, g^{m+s_1+s_2}) \ .$$

- Decryption: given a ciphertext $c = (c_1, c_2, c_3) \in \mathbb{G}_1^3$, output

$$\mathsf{Dec}_{\mathsf{sk}}(c_1, c_2, c_3) \leftarrow \log_g(c_3 \cdot c_2^{-1/(\mathsf{sk}+1)} \cdot c_1^{-1/\mathsf{sk}}) \ .$$

Note that decryption is well defined, since for all $m, s_1, s_2 \in \mathbb{Z}_p$,

$$\begin{aligned}
\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m; s_1, s_2)) &= \mathsf{Dec}_{\mathsf{sk}}(h^{s_1}, (gh)^{s_2}, g^m g^{s_1+s_2}) \\
&= \log_g(g^m g^{s_1+s_2} \cdot (gh)^{-s_2/(\mathsf{sk}+1)} \cdot h^{-s_1/\mathsf{sk}}) \\
&= \log_g(g^m g^{s_1+s_2} \cdot g^{-s_2} \cdot g^{-s_1}) \\
&= \log_g(g^m) \\
&= m \ .
\end{aligned}$$

2.10.3 Homomorphic Properties

Definition 10. Consider a probabilistic public key cryptosystem with message space $\mathcal{M}$ with group operation $*$, and ciphertext space $\mathcal{C}$ with group operation $\circ$. It is *homomorphic* [80] if for all $m_1, m_2 \in \mathcal{M}$, $\mathsf{Enc}_{\mathsf{pk}}(m_1) \circ \mathsf{Enc}_{\mathsf{pk}}(m_2) = \mathsf{Enc}_{\mathsf{pk}}(m_1 * m_2)$. If the operation $*$ is multiplication, then the cryptosystem is *multiplicatively homomorphic*. If the operation $*$ is addition, then the cryptosystem is *additively homomorphic*.

The ElGamal cryptosystem is multiplicatively homomorphic, since $\mathsf{Enc}_{\mathsf{pk}}(m_1; r_1) \cdot \mathsf{Enc}_{\mathsf{pk}}(m_2; r_2) = \mathsf{Enc}_{\mathsf{pk}}(m_1 \cdot m_2; r_1 + r_2)$. The ILin cryptosystem is additively homomorphic, since $\mathsf{Enc}_{\mathsf{pk}}(m_1; s_1, s_2) \cdot \mathsf{Enc}_{\mathsf{pk}}(m_2; s_1', s_2') = \mathsf{Enc}_{\mathsf{pk}}(m_1 + m_2; s_1 + s_1', s_2 + s_2')$.

2.11 Trapdoor Commitment Schemes

Definition 11. A *trapdoor commitment scheme* [46] is a cryptographic tool that consists of two algorithms:

- Commitment key generation algorithm gencom: given the security parameter κ, outputs a *commitment key* ck and a *trapdoor* (or *equivocation key*) td.

- Commitment algorithm Com: given a message $m \in \mathcal{M}$ and randomness $r \in \mathcal{R}$, outputs a *commitment* $c \leftarrow \mathsf{Com}_{\mathsf{ck}}(m; r)$. For $m \in \mathcal{M}$, define $\mathsf{Com}_{\mathsf{ck}}(m)$ to be the distribution of commitments of m over the commitment space.

The two algorithms must be efficient to compute, and satisfy the following security properties.

- *Perfectly hiding*: commitments to any two messages have the same distribution. That is, for any $m_0, m_1 \in \mathcal{M}$ and commitment key ck,

$$\mathsf{Com}_{\mathsf{ck}}(m_0) = \mathsf{Com}_{\mathsf{ck}}(m_1) \ .$$

  In this sense, a commitment algorithm "hides" the original message.

- *Computationally binding*: given a commitment, an adversary (without access to the trapdoor) has negligible probability of opening the commitment to two different messages. That is, for any NUPPT $\mathcal{A}$,

$$\Pr\begin{bmatrix}\mathsf{ck} \leftarrow \mathsf{gencom}(1^\kappa), (m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}(\mathsf{ck}) : \\ m_0 \neq m_1 \wedge \mathsf{Com}_{\mathsf{ck}}(m_0; r_0) = \mathsf{Com}_{\mathsf{ck}}(m_1; r_1)]\end{bmatrix} = \mathbf{negl}(\kappa) \ .$$

  In this sense, a commitment algorithm "binds" the original message to its commitment.

- *Trapdoor*: given access to the trapdoor, a commitment and the original message and randomness values used to compute it, one can open the commitment to any other message. That is, for any $m_0, m_1 \in \mathcal{M}$, there exists trapdoor opening algorithm that, given input $(\mathsf{ck}, \mathsf{td}; m_0, m_1; r)$, outputs a value r_{td} such that $\mathsf{Com}_{\mathsf{ck}}(m_0; r) = \mathsf{Com}_{\mathsf{ck}}(m_1; r_{\mathsf{td}})$, where the distribution of "normal" openings (m_0, r) and trapdoor openings (m_1, r_{td}) are identical.

Note that ElGamal can be viewed as a perfectly binding and computationally hiding (non-trapdoor) commitment scheme. Assume we work in a group $\mathbb{G}$ of prime order p, and given a generator $g \in \mathbb{G}$. The most well known trapdoor commitment scheme to commit to an integer $m \in \mathbb{Z}_p$ is the Pedersen commitment scheme [78] defined as follows.

- Commitment key generation: Generate $x \leftarrow_r \mathbb{Z}_p$, and set $h = g^x$. Output $\mathsf{ck} = (g, h)$ and $\mathsf{td} = x$.

- Commitment: Generate $r \leftarrow_r \mathbb{Z}_p$, and output $\mathsf{Com}_{\mathsf{ck}}(m; r) = g^m h^r$.

- Trapdoor: Given m, r and $c = \mathsf{Com}_{\mathsf{ck}}(m; r) = g^m h^r$, one can open to any $m' \in \mathbb{Z}_p$ using $r_{td} = r + \frac{m-m'}{x}$.

Using Pedersen, we can commit to a vector $\boldsymbol{m} \in \mathbb{Z}_p^n$ using n separate commitments. However, in some cases we want a commitment to a vector to be independent to the length of the vector. This can be done by the extended Pedersen commitment scheme [33] defined as follows.

- Commitment key generation: Generate $x, x_1, \cdots, x_n \leftarrow_r \mathbb{Z}_p$, and set $h = g^x, g_1 = g^{x_1}, \cdots, g_n = g^{x_n}$. Output $\mathsf{ck} = (g, h, g_1, \cdots, g_n)$ and $\mathsf{td} = (x, x_1, \cdots, x_n)$.
- Commitment: Generate $r \leftarrow_r \mathbb{Z}_p$, and output

$$\mathsf{Com}_{\mathsf{ck}}(\boldsymbol{m}; r) = h^r \prod_{i=1}^{n} g_i^{m_i} \quad .$$

- Trapdoor: Same as the Pedersen commitment scheme.

Using the extended Pedersen commitment scheme, a commitment of a vector $\boldsymbol{m} \in \mathbb{Z}_p^n$ is just one group element. Both the Pedersen and extended Pedersen commitment schemes are perfectly hiding and computationally binding under the DL assumption.

2.11.1 Polynomial Commitment Schemes

The extended Pedersen commitment scheme can be further generalized into a family of *polynomial commitment schemes* [56, 67] which we name PolyCommit. The trapdoor $\mathsf{td} = (x, x_1, \cdots, x_n)$ in extended Pedersen is replaced by a set of polynomials $(P_i(X))_{i=0}^n$ evaluated at a random point $X = \chi$, and the new trapdoor will then be $td = \chi$.

Definition 12. Assume we work in a group $\mathbb{G}$ of prime order p, and given a generator $g \in \mathbb{G}$. Let $(P_i(X))_{i=0}^n$ be a tuple of polynomials. The $(P_i(X))_{i=0}^n$-PolyCommit scheme in $\mathbb{G}$ can be defined as follows.

- Commitment key generation: Generate $\chi \leftarrow_r \mathbb{Z}_p$, and set $h = g^{P_0(\chi)}, h_1 = g^{P_1(\chi)}, \cdots, h_n = g^{P_n(\chi)}$. Output $\mathsf{ck} = (g, h, h_1, \cdots, h_n)$ and $\mathsf{td} = \chi$.
- Commitment: Generate $r \leftarrow_r \mathbb{Z}_p$, and output

$$\mathsf{Com}_{\mathsf{ck}}(\boldsymbol{m}; r) = h^r \prod_{i=1}^{n} h_i^{m_i} = g^{rP_0(\chi) + \sum_{i=1}^{n} m_i P_i(\chi)} \quad .$$

A general constraint in the choice of the polynomials $(P_i(X))_{i=0}^n$ is that they are linearly independent (otherwise, the binding property can easily be broken). Under a computational assumption, this is also sufficient to get a perfectly hiding and computationally binding commitment scheme. Some known choice of polynomials for PolyCommit are as follows.

- (Groth [56]) $P_i(X) = X^i$ for $i \in [0 \mathinner{..} n]$.

- (Lipmaa [67]) $P_i(X) = X^{\lambda_i}$ for a distinct set of integers $(\lambda_i)_{i=0}^n$.

- (Lipmaa [69]) Let ω be a primitive n-th root of unity modulo p, and $\omega_i = \omega^{i-1}$ for $i \in [1 \mathinner{..} n]$. Set $P_0(X) = \prod_{i=1}^n (X - \omega_i)$ and for $i \in [1 \mathinner{..} n]$ set $P_i(X)$ to be the Lagrange polynomial $P_i(X) = \prod\limits_{j \neq i} \frac{X-\omega_j}{\omega_i - \omega_j}$.

PolyCommit can also be defined in the bilinear group setting with $\mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$. Assume that g_1 is a generator of $\mathbb{G}_i$ and g_2 is a generator of $\mathbb{G}_2$. Then a commitment in $\mathbb{G}_1$ will be of the form

$$c = g_1^{rP_0(\chi) + \sum\limits_{i=1}^n m_i P_i(\chi)} .$$

Assuming the polynomials $(P_i(X))_{i=0}^n$ are linearly independent, $(P_i(X))_{i=0}^{n+1}$-PolyCommit is perfectly hiding and computationally binding under the $(P_i(X))_{i=0}^{n+1}$-PDL assumption [41].

To guarantee that the committer knows what vector he commits to, we can additionally add a *knowledge component* of the form

$$\tilde{c} = (g_2^\gamma)^{rP_0(\chi) + \sum\limits_{i=1}^n m_i P_i(\chi)}$$

for a knowledge secret γ. Under the PKE assumption, if the commitment is done in $\mathbb{G}_1$ with a knowledge component in $\mathbb{G}_2$ (or vice versa), there exists an extractor that can extract the committed vector [55].

2.12 Zero-knowledge Proofs

Let $\mathcal{R}$ be a binary relation, and let $\mathcal{L} = \{x \mid \exists w : (x, w) \in \mathcal{R}\}$ be a group-dependent language. It is usually assumed that $\mathcal{L} \in \mathbf{NP}$, hence one can efficiently verify whether or not $(x, w) \in \mathcal{R}$. Let $\bar{\mathcal{L}} = \{x \mid x \notin \mathcal{L}\}$.

In an *interactive proof* [51], a prover tries to convince a verifier, by an exchange of messages, that a statement x is in the language $\mathcal{L}$. An interactive proof has two properties, as follows.

- Completeness: if $x \in \mathcal{L}$, an honest verifier will accept the prover's proof.

- Soundness: if $x \in \bar{\mathcal{L}}$, a malicious prover can not provide a convincing proof to an honest verifier.

A common three round interactive proof used in cryptography is a *sigma protocol* [9], where the three messages are known as commitment, challenge, and

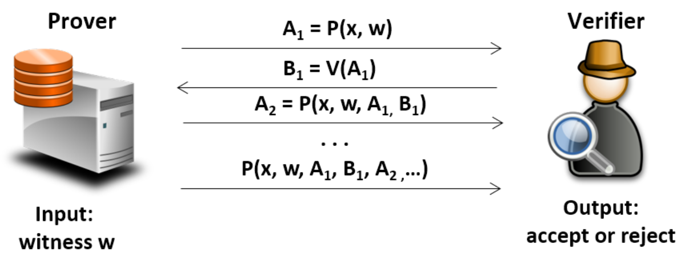

Figure 2.1: Interactive proof illustration. The prover and verifier exchange messages until the verifier either accepts or rejects the proof. Previous messages in the interaction may be used to create subsequent ones.

response. If the proof consists of just one message from the prover to the verifier, we get a *non-interactive proof* [32].

An easy way to achieve a short (and non-interactive) proof is for the prover to send w to the verifier, where the verifier can quickly check whether $(x, w) \in \mathcal{R}$. However, this will give away the witness w, meaning the verifier can later prove the same statement x to another verifier.

On the other extreme, we would like to prove a statement $x \in \mathcal{L}$ without giving anything away to the verifier about the value w, besides that which can be derived without interacting with the prover. A proof with such a property is known as a *zero-knowledge proof* [54]. If soundness is guaranteed only for NUPPT algorithms, then we instead get a *zero-knowledge argument*.

There are two common models to make an argument non-interactive.

- *Random oracle* (RO) model [50]. In this model, we assume the existence of one or more random oracles that respond to every (unique) query with a uniformly random value from its output domain. Both the prover and verifier can access these random oracles. The RO model makes use of the fact that an honest verifier's messages to the prover should be uniformly random, and do not depend on the prover's messages. Using the so-called Fiat-Shamir heuristic [45], each message the verifier needs to send in an argument is omitted, and is instead replaced by a random oracle query. The argument thus becomes a single message from the prover to the verifier.

- *Common reference string* (CRS) model [23]. In the CRS model, we assume the existence of a trusted party that can generate a common string that can be referenced by both the prover and verifier. This common reference string in a way incorporates part of an honest verifier's reply to the prover, and is independent to the statement to be proven. It must be correctly generated so that the proof can be complete, sound, and zero-knowledge. The CRS is only generated once, but can be used for multiple interactions.

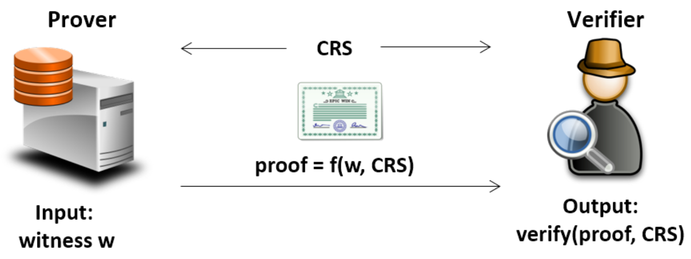

Figure 2.2: Non-interactive zero-knowledge proof in the CRS model.

Security proofs in the CRS model are more convincing than the Fiat-Shamir heuristic used in the RO model. The Lindell [66] and Ciampi et al. [29] transforms can be used to improve heuristic security compared to the Fiat-Shamir heuristic, but they use both RO and CRS.

Moreover, there are two models which can be seen as relaxations of the CRS model: the registered public key (RPK) model, and the bare public key (BPK) model.

- In the RPK model [4], a verifier's randomness is incorporated into a public key that is registered in a central authority. The public key can either be generated randomly by the central authority, or generated by the verifier itself. In the latter case, the verifier must send the private data related to the public key to a central authority, who then checks that it meets some set requirements. A prover who wants to send a proof to a verifier, must first query the verifier's public key from the central authority, and from that generate the proof.

- In the BPK model [24], a verifier's randomness is again incorporated into a public key, but differently to the RPK model, this public key need not be checked by a central authority. In this model, extra steps are needed to guard against a dishonest verifier who generates a malicious public key.

Note that in both cases, the non-interactive proof can only be verified if one knows the secret key of the designated verifier, and hence is only valid and checkable by a single verifier. This means that NIZK arguments in the BPK and RPK models are not transferable. While this in itself is not a bad thing, it can becomes a problem if one wants to prove the same statement to several different verifiers.

We now define non-interactive zero-knowledge arguments in the CRS model. The following is a high-level version of the formal definitions given by Fauzi and Lipmaa [39].

Definition 13. Let $\mathcal{R}$ be a binary relation, and let $\mathcal{L} = \{x \mid \exists w : (x, w) \in R\}$ be a group-dependent language. A *non-interactive zero-knowledge argument* [54] in the CRS model consists of four algorithms.

- Setup algorithm $\mathsf{BP}(1^\kappa, n)$: generates the bilinear group description $\mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$.
- CRS generation algorithm $\mathsf{gencrs}(\mathsf{gk})$: generates the common reference string crs and a trapdoor td. Although the prover and verifier can access the same crs, it may only require part of it to preform its computation. The part that the prover needs to compute a proof is denoted crs_p, while the part that the verifier needs to verify a proof is denoted crs_v. This separation of the CRS is only needed for efficiency.
- Prover pro: outputs an argument $\pi = \mathsf{pro}(\mathsf{gk}, \mathsf{crs}_p; x, w)$.
- Verifier ver: outputs $\mathsf{ver}(\mathsf{gk}, \mathsf{crs}_v; x, \pi)$ which is either accept or reject.

The desired properties of an argument are as follows.

- *Perfectly complete*: if the prover is honest and $x \in \mathcal{L}$, an honest verifier always accepts prover's proof. Formally,

$$\Pr\left[\begin{matrix}\mathsf{gk} \leftarrow \mathsf{BP}(1^\kappa, n), ((\mathsf{crs}_p, \mathsf{crs}_v), \mathsf{td}) \leftarrow \mathsf{gencrs}(\mathsf{gk}), \\ (x, w) \leftarrow \mathcal{R}(\mathsf{gk}) : \mathsf{ver}(\mathsf{gk}, \mathsf{crs}_v; x, \mathsf{pro}(\mathsf{gk}, \mathsf{crs}_p; x, w)) = 1\end{matrix}\right] = 1 \ .$$

- *Adaptively computationally sound*: for any NUPPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{matrix}\mathsf{gk} \leftarrow \mathsf{BP}(1^\kappa, n), ((\mathsf{crs}_p, \mathsf{crs}_v), \mathsf{td}) \leftarrow \mathsf{gencrs}(\mathsf{gk}), \\ (x, \pi) \leftarrow \mathcal{A}(\mathsf{gk}, \mathsf{crs}_p, \mathsf{crs}_v) : x \in \bar{\mathcal{L}} \ \wedge \\ \mathsf{ver}(\mathsf{gk}, \mathsf{crs}_v; x, \pi) = 1\end{matrix}\right] = \mathbf{negl}(\kappa) \ .$$

  Here, an adversary is *adaptive* in the sense that it can use the CRS in its attempt to construct a proof of an incorrect statement.

- *Perfectly zero-knowledge*: there exists a simulator sim, with access to the trapdoor td and without knowing the witness w, that can output a proof π' such that $(\mathsf{gk}, x, \mathsf{crs}, w, \pi')$ has the same distribution as $(\mathsf{gk}, x, \mathsf{crs}, w, \pi)$, where $\pi = \mathsf{pro}(\mathsf{gk}, \mathsf{crs}_p; x, w)$ is the proof of an honest prover. That is, there exists a probabilistic polynomial-time algorithm sim such that for any $(x, w) \in \mathcal{R}$ and any stateful non-uniform adversary $\mathcal{A}$,

$$\Pr\left[\begin{matrix}\mathsf{gk} \leftarrow \mathsf{BP}(1^\kappa, n), \\ (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{gencrs}(\mathsf{gk}), \\ \pi \leftarrow \mathsf{pro}(\mathsf{gk}, \mathsf{crs}_p; x, w) : \\ \mathcal{A}(\mathsf{gk}, x, \mathsf{crs}, w, \pi) = 1\end{matrix}\right] = \Pr\left[\begin{matrix}\mathsf{gk} \leftarrow \mathsf{BP}(1^\kappa, n), \\ (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{gencrs}(\mathsf{gk}), \\ \pi \leftarrow \mathsf{sim}(\mathsf{gk}, \mathsf{crs}; x, \mathsf{td}) : \\ \mathcal{A}(\mathsf{gk}, x, \mathsf{crs}, w, \pi) = 1\end{matrix}\right] \ .$$

Essentially, this means that an accepting proof could have been generated by a simulator that doesn't know the witness, hence the verifier learns nothing from the proof besides the correctness of the statement.

Note that the prover and simulator use the same CRS, in which case we have *same-string* zero-knowledge [31]. De Santis et al. [31] proved that the notion of perfect and same-string zero-knowledge enables the CRS to be used an unbounded number of times to create arguments, without losing the perfect zero-knowledge property. Moreover, Abe and Fehr [1] proved that it is possible to create an argument for any $\mathbf{NP}$ language that is both adaptively sound and perfectly zero-knowledge under the above definition.

Additionally, an argument can have the following properties. Let $\mathcal{R}^*$ be the set of all relations $\mathcal{R}_{\mathsf{guilt}}$ which are decidable in polynomial time and consist of pairs (x, w_{guilt}) such that $x \notin \mathcal{L}$ is an incorrect statement, and w_{guilt} is a witness that x is an incorrect statement. For $\mathcal{R}_{\mathsf{guilt}} \in \mathcal{R}^*$, we can define $\mathcal{L}_{\mathsf{guilt}} = \{x \mid \exists w_{\mathsf{guilt}} : (x, w_{\mathsf{guilt}}) \in \mathcal{R}_{\mathsf{guilt}}\}$

- *Adaptive culpable soundness* [59]: for any binary relation $\mathcal{R}_{\mathsf{guilt}} \in \mathcal{R}^*$ and any NUPPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{l} \mathsf{gk} \leftarrow \mathsf{BP}(1^\kappa, n), ((\mathsf{crs}_p, \mathsf{crs}_v), \mathsf{td}) \leftarrow \mathsf{gencrs}(\mathsf{gk}), \\ (x, \pi, w_{\mathsf{guilt}}) \leftarrow \mathcal{A}(\mathsf{gk}, \mathsf{crs}_p, \mathsf{crs}_v) : \\ (x, w_{\mathsf{guilt}}) \in \mathcal{R}_{\mathsf{guilt}} \ \wedge \mathsf{ver}(\mathsf{gk}, \mathsf{crs}_v; x, \pi) = 1 \end{array}\right] = \mathbf{negl}(\kappa) \ .$$

  Note that for any choice of $\mathcal{R}_{\mathsf{guilt}}$ we have that $\mathcal{L}_{\mathsf{guilt}} \subseteq \bar{\mathcal{L}}$, so culpable soundness is a weaker security notion than adaptive soundness.

- *Perfectly witness-indistinguishable* [44]: if $(x, w_0) \in \mathcal{R}$ and $(x, w_1) \in \mathcal{R}$ for the same statement x, then $\mathsf{pro}(\mathsf{gk}, \mathsf{crs}_p, x, w_0)$ and $\mathsf{pro}(\mathsf{gk}, \mathsf{crs}_p, x, w_1)$ have the same distribution. Essentially, this means that given a valid proof π, a verifier cannot know which of the valid witnesses was used to generate the proof.

- *Argument of knowledge* [43]: if the verifier accepts a proof π, then there exists an extractor that extracts the witness w from π. That is, for any NUPPT adversary $\mathcal{A}$, there exists a NUPPT algorithm $X_{\mathcal{A}}$ such that

$$\Pr\left[\begin{array}{l} \mathsf{gk} \leftarrow \mathsf{BP}(1^\kappa, n), ((\mathsf{crs}_p, \mathsf{crs}_v), \mathsf{td}) \leftarrow \mathsf{gencrs}(\mathsf{gk}), \\ ((x, \pi); w) \leftarrow (\mathcal{A}||X_{\mathcal{A}})(\mathsf{crs}_p, \mathsf{crs}_v; \mathsf{aux}) : \\ (x, w) \notin \mathcal{R} \wedge \mathsf{ver}(\mathsf{gk}, \mathsf{crs}_v; x, \pi) = 1 \end{array}\right] = \mathbf{negl}(\kappa) \ .$$

  As in the PKE assumption in Section 2.8, we assume aux to be a common auxiliary input to Adv and X_{Adv} that is generated by using benign auxiliary input generation. Essentially, this means that a valid proof π shows that the prover *knows* the witness w it used to generate the proof.

2.13 Techniques for NIZK in the CRS model

In this section we will briefly discuss some known techniques that can be used to construct efficient NIZK arguments in the CRS model. We will use Square Span Programs in the construction of shuffle arguments in Section . Although we do not use Groth-Sahai proofs in this thesis, we will provide a brief overview as it is a well-known technique.

2.13.1 Square Span Programs

Square span programs were defined by Danezis et al. [30] to provide a means of checking arithmetic circuits more efficiently.

Definition 14. A *square span program* (SSP) Q of size m and degree d over the field $\mathbb{Z}_q$ is a collection of $m+1$ polynomials $v_0(X), v_1(X), \ldots, v_m(X)$ of degree at most d and a target polynomial $t(X)$ of degree d. We say that Q accepts an input $\boldsymbol{a} \in \mathbb{Z}_q^\ell$ iff there exist integers $a_{\ell+1}, \cdots, a_m \in \mathbb{Z}_q$ satisfying

$$t(X) \mid \left((v_0(X) + \sum_{i=1}^{m} a_i v_i(X))^2 - 1 \right) \quad .$$

SSP can be seen as a generalization of quadratic span programs (QSP) introduced by Gennaro et al. [49], where checking whether or not a vector $\boldsymbol{a}$ satisfies some properties is equivalent to checking if the target polynomial divides $\sum_{i=1}^{m} a_i v_i(X))^2 - 1$. Since this can be done using only one polynomial evaluation, SSP leads to very efficient checking of arithmetic circuits.

2.13.2 Groth-Sahai Proofs

Groth and Sahai [60] defined a set of techniques, known as *Groth-Sahai proofs*, to achieve NIZK in the CRS model for a large family of algebraic equations. In a Groth-Sahai proof, the prover wants to prove that there are values $\boldsymbol{x}, \boldsymbol{y}$ that satisfy a set of equations simultaneously, while keeping these values private.

Let $\mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$ be a bilinear group description. Groth and Sahai defined four types of algebraic equations as follows.

- *Pairing product equation.*

$$\prod_{i=1}^{m} \hat{e}(x_i, b_i) \cdot \prod_{j=1}^{n} \hat{e}(a_j, y_j) \cdot \prod_{i=1}^{m} \prod_{j=1}^{n} \hat{e}(x_i, y_j)^{\gamma_{ij}} = t_T \quad ,$$

where $x_i \in \mathbb{G}_1$, $y_j \in \mathbb{G}_2$ are private and $a_i \in \mathbb{G}_1$, $b_j \in \mathbb{G}_2$, $\gamma_{ij} \in \mathbb{Z}_p$ and $t_T \in \mathbb{G}_T$ are public constants.

- *Multiscalar multiplication equation* in $\mathbb{G}_1$.

$$\prod_{i=1}^{m} x_i^{b_i} \cdot \prod_{j=1}^{n} a_j^{y_j} \cdot \prod_{i=1}^{m}\prod_{j=1}^{n} x_i^{\gamma_{ij} y_j} = t_1 \ ,$$

  where $x_i \in \mathbb{G}_1$, $y_j \in \mathbb{Z}_p$ are private and $a_j \in \mathbb{G}_1$, $b_i \in \mathbb{Z}_p$, $\gamma_{ij} \in \mathbb{Z}_p$ and $t_1 \in \mathbb{G}_1$ are public constants.

- *Multiscalar multiplication equation* in $\mathbb{G}_2$.

$$\prod_{i=1}^{m} b_i^{x_i} \cdot \prod_{j=1}^{n} y_j^{a_j} \cdot \prod_{i=1}^{m}\prod_{j=1}^{n} y_j^{\gamma_{ij} x_i} = t_2 \ ,$$

  where $x_i \in \mathbb{Z}_p$, $y_j \in \mathbb{G}_2$ are private and $a_j \in \mathbb{Z}_p$, $b_i \in \mathbb{G}_2$, $\gamma_{ij} \in \mathbb{Z}_p$ and $t_2 \in \mathbb{G}_2$ are public constants.

- *Quadratic equation* in $\mathbb{Z}_p$.

$$\sum_{i=1}^{m} x_i b_i + \sum_{j=1}^{n} a_j y_j + \sum_{i=1}^{m}\sum_{j=1}^{n} \gamma_{ij} x_i y_j = t \ ,$$

  where $x_i, y_j \in \mathbb{Z}_p$, are private and $a_j, b_i, \gamma_{ij}, t \in \mathbb{Z}_p$ are public constants.

The main idea of a Groth-Sahai proof is transforming equations involving private variables in the message space into corresponding equations in the commitment space which hides the private variables. Under falsifiable [75] security assumptions (where we can check in polynomial time if an adversarial strategy breaks the security assumption), if the equations hold in the commitment space, then with overwhelming probability the corresponding equations hold in the message space.

Assume that there are K equations involving the same $L = m + n$ private variables $(x_i)_{i \in [1\,..\,m]}$ and $(y_j)_{j \in [1\,..\,n]}$. Then the Groth-Sahai proof will require the prover to compute and send $\Theta(L)$ commitments and $\Theta(K)$ proofs, and the verifier has to check $\Theta(K)$ equations. In the case of a pairing product equation, the prover will have to compute $\Theta(K+L)$ group exponentiations while the verifier has to compute $\Theta(K + L)$ pairings.

Groth-Sahai proofs achieve good asymptotic efficiency, but the specific constants involved are not small enough in practice. Escala and Groth [37] constructed Groth-Sahai proofs with smaller constants, but it is still not optimal. For example, in Section 5 we will see several NIZK shuffle arguments that are more efficient than the existing construction based on Groth-Sahai proofs.

CHAPTER 3

MODULAR NIZK ARGUMENTS

3.1 Motivation

Many cryptographic protocols require a certain party to provide a proof that they have expended significant amounts of resources (e.g., CPU power) before sending a message. This is known as a proof-of-work. For example, Bitcoin uses the Hashcash proof-of-work system [3] where the "work" consists of finding a string x such that its hash $H(x)$ starts with k consecutive zeroes. A typical application is is spam e-mail filtering, where an e-mail server will filter out messages that are not accompanied with a proof-of-work. There are two main models for showing proof-of-work: challenge-response and verifiable computation.

In the challenge-response model, a challenger gives some challenge message m to the prover, who must in turn respond with a correct message which solves the challenge. The challenger then verifies the response, and only proceeds in the protocol execution if verification succeeds. This model is interactive, which is not ideal in many scenarios. For example, in the spam e-mail filtering scenario, an e-mail sender need not be online after sending the e-mail.

Meanwhile, in the verifiable computation model [48], a prover both chooses and solves a certain hard problem, and sends both the problem (in the form of a statement) and solution (in the form of a witness) to the verifier. This can be simplified by agreeing on a well-known hard language $\mathcal{L}$, with easy-to-verify relation $\mathcal{R}$ that consists of all statement-witness pairs (typically $\mathbf{NP}$-complete languages), where the prover sends a statement x along with a corresponding witness w. The verifier then only needs to verify that $x \in \mathcal{L}$ and that $(x, w) \in \mathcal{R}$. In this model, the verifier may not have the resources to do large amounts of computation (such as finding the correct witness w himself), so ideally verification does not require much computation. The prover need not be online during verification.

Since we want a proof-of-work without interaction, the preferred model is verifiable computation for $\mathbf{NP}$. However, in some cases of verifiable computation, we also want to hide the value w, since the verifier might be able to use it maliciously (to breach privacy, construct his own (x', w') pair for another protocol execution, etc.). However, we still want to be able to verify that the computation was done correctly. To solve this, we need to construct a NIZK argument for $\mathbf{NP}$ languages.

In a typical NIZK argument, a prover commits to the witness w, and proves that the committed values satisfy some desired properties. Many $\mathbf{NP}$ languages require that the witness is from a certain range of values (e.g., a solution to a 0-1 Integer Linear Programming problem must be a binary vector). Hence a prover must additionally prove that he commits to a witness w from the correct range, without revealing anything about w. This is known as a range proof [18, 70, 85, 26]. In this case, a range proof will be a building block in constructing the full NIZK argument for that $\mathbf{NP}$ language. However, as we will discuss in the next chapter, a range proof can be useful in other scenarios.

3.2 Previous Work

One of the earliest work that constructed a CRS-based NIZK argument for $\mathbf{NP}$ languages was by Groth [56], which has two interesting properties. Firstly, it is succinct, meaning the proof size is constant, and the verification time is short (at most linear in the size of the witness). Secondly, it is modular, meaning that he first constructs some basic NIZK arguments, including the product and permutation arguments, which are then used as building blocks to construct a NIZK argument for CIRCUIT-SAT. The product argument proves that a committed vector is a Hadamard product of two other committed vectors, and the permutation argument proves that a committed vector is a permutation of another committed vector.

The advantage of a modular approach is that any improvements in the basic arguments will propagate to the main protocol. Moreover, the basic arguments can also be used to get NIZK arguments for other languages (e.g. shuffle in [71], see Section 5 for further discussion on shuffle) or other $\mathbf{NP}$ languages (see [69] for a list). For CIRCUIT-SAT, complexity is defined through the size n of the circuit. Groth achieved constant communication, CRS size of $\Theta(n^2)$ group elements, prover's computation of $\Theta(n^2)$ exponentiations, and verifier's computation of $\Theta(n)$ group multiplications and $\Theta(1)$ pairings. However, if the circuit size is large, then $\Theta(n^2)$ prover's computation is not ideal in practice.

Lipmaa [67] improved the CRS size and prover's computation of Groth's basic product and permutation arguments. This was done mostly by modifying Groth's polynomial commitment scheme, which if done correctly used $n + 1$ polynomials of degree up to n^2, to instead use $n + 1$ polynomials whose degrees are progression-free. This is an asymptotic improvement since by Corollary 1, $r_3^{-1}(n) = n^{1+o(1)}$. Since the basic arguments are more efficient, the resulting argument for CIRCUIT-SAT is also more efficient. However, the verifier's computation is slightly less efficient: $\Theta(n)$ exponentiations and $\Theta(1)$ pairings. Moreover, it still suffered the same drawback as Groth's scheme, i.e., the prover's computation is still $\Theta(n^2)$.

Gennaro et al. [49] introduced a technique that generalized the techniques used by Groth [56] and Lipmaa [67] into a new construction, which they named quadratic span programs (QSP).

Using a variant of QSP that efficiently computes arithmetic circuits, Gennaro et al. constructed an asymptotically efficient NIZK argument for CIRCUIT-SAT that has CRS size of $O(n)$ group elements and $n^{1+o(1)}$ prover's computation. Differently from the previously mentioned constructions, the Gennaro et al. NIZK argument for CIRCUIT-SAT is not modular.

Chaabouni, Lipmaa and Zhang [26] constructed a modular range argument (i.e. prove that a committed value is in some range $[A \mathinner{..} B]$) using Lipmaa's basic product and permutation arguments [67]. The range argument has $\Theta(h)^2$ prover's computation and $\Theta(h)$ verifier's computation, where $h = \log_2(B - A)$.

3.3 Problem Statement

Can we construct a NIZK range argument and NIZK arguments for $\mathbf{NP}$-complete languages that is succinct but with sub-quadratic prover's complexity? Moreover, can we do it in a modular way?

3.4 Our Solution

We start with two basic succinct arguments, product and shift, and use them to construct NIZK range argument and NIZK arguments for $\mathbf{NP}$-complete languages in a modular way. Since the basic arguments are succinct, and in each constructed argument we only use a constant number of basic arguments, all constructed argument are thus also succinct. A full description of our construction is given in the paper [41], which is a joint work with Lipmaa and Zhang.

3.4.1 Committing to a Vector

Let $S = (\lambda_1, \cdots, \lambda_n)$ be progression-free with $\lambda_{i+1} > \lambda_i$, and let $\upsilon > 2\lambda_n - \lambda_1$. We use the $(P_i(X))_{i=0}^n$-PolyCommit scheme of Section 2.11.1 with $P_i(X) = X^{\lambda_i}$ for $i \in [1 \mathinner{..} n]$, and $P_0(X) = X^\upsilon$. We also generate a knowledge secret

$\gamma \leftarrow_r \mathbb{Z}_p$. Then

$$\mathsf{Com}(\boldsymbol{a}; r) = (g_1, g_1^{\gamma})^{r\sigma^{\upsilon} + \sum_{i=1}^{n} a_i \sigma^{\lambda_i}} \enspace .$$

For a commitment $\mathcal{C} = (C, \tilde{C})$, the value $\tilde{C}$ is known as the knowledge component and is necessary to ensure the committer actually knows what vector he commits to. This concrete choice of polynomials was first introduced by Lipmaa [67], but instead of Lipmaa's choice of $\upsilon = 0$, we instead use a general $\upsilon > 2\lambda_n - \lambda_1$.

3.4.2 Product Argument

In a product argument [56], we aim to show that three commitments

$$\begin{aligned} \mathcal{A} &= \mathsf{Com}(\boldsymbol{a}; r_a) \enspace , \\ \mathcal{B} &= \mathsf{Com}(\boldsymbol{b}; r_b) \enspace , \quad \text{and} \\ \mathcal{C} &= \mathsf{Com}(\boldsymbol{c}; r_c) \end{aligned}$$

can be opened to $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$, such that the three vectors satisfy the Hadamard product $\boldsymbol{c} = \boldsymbol{a} \circ \boldsymbol{b}$. The construction uses pairings and the fact that the set $S = (\lambda_1, \cdots, \lambda_n)$ used in PolyCommit is progression-free. Full details can be seen in [41].

The resulting product argument is similar to the one by Lipmaa [67]. However, we make it even more efficient by using FFT instead of polynomial multiplication to compute $\log_{g_2} \pi$ as a polynomial. This reduces the prover's computation from $\Theta(n^2)$ to $\Theta(N \log N)$, where $N = r_3^{-1}(n)$. This is an improvement since $N \log N = n^{1+o(1)}$.

3.4.3 Shift, Rotation and Scan Arguments

In a shift-by-η argument, we aim to show that two commitments

$$\begin{aligned} \mathcal{A} &= \mathsf{Com}(\boldsymbol{a}; r_a) \ \text{and} \\ \mathcal{B} &= \mathsf{Com}(\boldsymbol{b}; r_b) \end{aligned}$$

can be opened to $\boldsymbol{a}$, $\boldsymbol{b}$, where $\boldsymbol{a}$ is a right shift-by-η of $\boldsymbol{b}$. In other words, $a_i = b_{i+\eta}$, where we define $b_i = 0$ when $i > n$. Similarly, we construct a rotation-by-η argument where the vectors satisfy $a_i = b_{i+\eta}$ if $i + \eta \le n$, and $a_i = b_{i+\eta-n}$ otherwise. We also construct a scan argument, where $\boldsymbol{b}$ is a scan of $\boldsymbol{a}$ if $b_i = \sum_{j=i+1}^{n} a_j$. The scan argument is interesting as it enables to provide proofs that involve the sum of elements of a committed vector. As we will see below, this is needed as an intermediate step in various NIZK arguments for $\mathbf{NP}$.

The shift and rotation arguments are simpler than the permutation argument used in previous work [56, 67], and also much more efficient. For soundness of the shift and rotation arguments, we need the value $\upsilon > \lambda_n + \eta$, which for $\eta \in [1 \mathinner{..} n]$ always holds when $\upsilon > 2\lambda_n - \lambda_1$.

3.4.4 Range Argument

Using the product and shift arguments, we construct a more efficient range argument than the one by Chaabouni et al. [26]. The efficiency gain is mostly from rewriting the Chaabouni et al. argument to work without the permutation argument, and only using the more efficient product and shift arguments. This is possible because the permutation argument was essentially used to construct a scan argument, which we can implement by just one shift-by-1 argument.

The resulting range argument is much simpler than the Chaabouni et al. range argument, since we do not require a permutation argument. Since the commitment scheme is additively homomorphic, we can assume that the interval for the range proof is $[0 \mathinner{..} H]$. As in [26], we use the fact that $x \in [0 \mathinner{..} H]$ iff there exist values $(b_i)_{i=1}^{\|H\|}$ such that $b_i \in \{0, 1\}$ and

$$x = \sum_{i=1}^{\|H\|} \lfloor \frac{H + 2^{i-1}}{2^i} \rfloor b_i \enspace . \tag{3.1}$$

We can prove that the Equation 3.1 holds using a restriction argument (that shows the first element of a vector is zero), a scan argument and a product argument.

3.4.5 NIZK Arguments for NP

Using the product and shift arguments, we can also construct NIZK arguments for any language in $\mathbf{NP}$. In our work, we illustrate NIZK arguments for three different $\mathbf{NP}$-complete languages. Security of the various arguments for $\mathbf{NP}$ follow from the security of the basic arguments. See Appendix E and Appendix F in [41] for detailed security proofs.

SET-PARTITION

In the SET-PARTITION problem, we are given a public multiset $\mathcal{S} = \{s_1, \cdots, s_n\}$ of integers in $\mathbb{Z}_p$ and must partition it into two sets $\mathcal{X}, \mathcal{Y}$ with the same sum modulo p. This is equivalent to finding $\boldsymbol{b} \in \{-1, 1\}^n$ such that $\sum_{i=1}^n b_i s_i \equiv 0 \pmod{p}$. Hence in the SET-PARTITION argument, a prover must provide a commitment $\mathcal{B}$ and prove it can be opened to a vector $\boldsymbol{b}$ that satisfies the conditions. The complete SET-PARTITION argument is as follows.

Algorithm 1: SET-PARTITION argument

```
Compute a product argument $\pi_1$ for $b_i \cdot b_i = 1$, showing that $b_i \in \{-1, 1\}$;
Compute a product argument $\pi_2$ showing that $c_i = b_i \cdot s_i$ for $i \in [1 \,..\, n]$;
Compute a scan argument $\pi_3$ showing that $\boldsymbol{d}$ is the scan of $\boldsymbol{c}$;
Compute a restriction argument $\pi_4$ showing that the first coordinate of $\boldsymbol{c} + \boldsymbol{d}$ is 0;
Let $B$ be a commitment to $\boldsymbol{b}$, $C$ be a commitment to $\boldsymbol{c}$, and $D$ be a commitment to
$\boldsymbol{d}$. The SET-PARTITION argument is equal to $(B, C, D, \pi_1, \dots, \pi_4)$;
```

SUBSET-SUM

In the SUBSET-SUM problem, we are given a public multiset $\mathcal{S} = \{s_1, \cdots, s_n\}$ of integers in $\mathbb{Z}_p$ and must find a non-empty subset $\mathcal{T}$ that sums to $0 \pmod{p}$. This is equivalent to finding $\boldsymbol{b} \in \{0,1\}^n$ such that $\sum_{i=1}^{n} b_i s_i \equiv 0 \pmod{p}$ and $\boldsymbol{b} \neq \mathbf{0}_n$. Note that as mentioned in Section 3.4.4, $\boldsymbol{b} \in \{0,1\}^n$ can be proven using a product argument. Moreover, $\sum_{i=1}^{n} b_i s_i = 0$ can be checked as in the SET-PARTITION argument. The only difference is that we need to prove that $\boldsymbol{b} \neq \mathbf{0}_n$, which can be done using Lipmaa and Zhang's zero argument [71]. The complete SUBSET-SUM argument is as follows.

Algorithm 2: SUBSET-SUM argument

```
Compute a product argument $\pi_1$ for $b_i^2 = b_i$, showing that $\boldsymbol{b}$ is Boolean;
Compute an argument $\pi_2$ showing that $\boldsymbol{b} \neq \mathbf{0}$;
Compute a product argument $\pi_3$ showing that $c_i = b_i \cdot s_i$ for $i \in [1 \,..\, n]$;
Compute a scan argument $\pi_4$ showing that $\boldsymbol{d}$ is the scan of $\boldsymbol{c}$;
Compute a restriction argument $\pi_5$ showing that the first coordinate of $\boldsymbol{c} + \boldsymbol{d}$ is 0;
Let $B$ be a commitment to $\boldsymbol{b}$, $C$ be a commitment to $\boldsymbol{c}$, and $D$ be a commitment to
$\boldsymbol{d}$. The SUBSET-SUM argument is equal to $(B, C, D, \pi_1, \dots, \pi_5)$;
```

DECISION-KNAPSACK

In the DECISION-KNAPSACK problem, we are given a public set $\mathcal{S} = [1 \,..\, n]$ of item indices, a public set $\mathcal{V} = \{v_1, \cdots, v_n\} \in \mathbb{N}^n$ of item values, and a public set $\mathcal{W} = \{w_1, \cdots, w_n\} \in \mathbb{N}^n$ of item weights. We are also given a target value $V \in \mathbb{N}$ and a weight limit $W \in \mathbb{N}$. From these given parameters, must decide whether or not there exists a set $\mathcal{T} \subseteq \mathcal{S}$ such that $\sum_{i \in \mathcal{T}} v_i \geq V \wedge \sum_{i \in \mathcal{T}} w_i \leq W$. The DECISION-KNAPSACK argument can be constructed using the range and SUBSET-SUM arguments, see Appendix F in [41] for a detailed construction.

3.5 Comparison with Previous Work

Similar to the NIZK arguments of Groth and Lipmaa, our NIZK arguments for $\mathbf{NP}$ languages are succinct. However, unlike both mentioned work we achieve $\Theta(N \log N) = n^{1+o(1)}$ prover's complexity while still comparable with Groth's verifier's complexity (i.e., $\Theta(n)$ group multiplications and $\Theta(1)$ pairings). If we extend the comparison to non-modular NIZK arguments, Gennaro et al. [49] had a slightly better prover's complexity of $n \cdot \log n^3$ multiplications in $\mathbb{Z}_p$, $\Theta(n)$ group multiplications and $\Theta(n)$ group exponentiations. Groth, Lipmaa, and Gennaro et al. all chose CIRCUIT-SAT as their sample $\mathbf{NP}$-complete language, while we use the SET-PARTITION, SUBSET-SUM and DECISION-KNAPSACK problems.

We summarize this comparison in Table 3.1. Note that since the choice of $\mathbf{NP}$ language affects the constants, we compare asymptotic efficiency and not the specific constants. Moreover, in the prover's and verifier's complexity, we omit addition and multiplication in $\mathbb{Z}_p$ and only consider group multiplications, exponentiations, and pairings. We emphasize that although the NIZK arguments for $\mathbf{NP}$-complete languages are not directly comparable, we demonstrate good efficiency for *some* $\mathbf{NP}$-complete languages.

Table 3.1: A comparison of our NIZK arguments for $\mathbf{NP}$ [41] with the Groth [56], Lipmaa [67] and Gennaro et al. [49] arguments in the CRS model. In the prover's and verifier's complexity, we omit addition and multiplication in $\mathbb{Z}_p$ and only consider group multiplications ($\mathfrak{m}$), exponentiations ($\mathfrak{e}$), and pairings ($\mathfrak{p}$). The best efficiency among the modular NIZK arguments are highlighted in bold. Note that $N = r_3^{-1}(n) = n^{1+o(1)}$.

	Groth	Lipmaa	Our result	Gennaro et al.
|CRS|	$\Theta(n^2)$	$\mathbf{\Theta(N)}$	$\mathbf{\Theta(N)}$	$\Theta(n)$
Comm.	$\mathbf{\Theta(1)}$	$\mathbf{\Theta(1)}$	$\mathbf{\Theta(1)}$	$\Theta(1)$
Prover	$\Theta(n^2)\mathfrak{e}$	$\Theta(N)\mathfrak{e}$	$\mathbf{\Theta(N)}\mathfrak{m}$	$\Theta(n)\mathfrak{m} + \Theta(n)\mathfrak{e}$
Verifier	$\mathbf{\Theta(n)}\mathfrak{m} + \mathbf{\Theta(1)}\mathfrak{p}$	$\Theta(n)\mathfrak{e}$	$\mathbf{\Theta(n)}\mathfrak{m} + \mathbf{\Theta(1)}\mathfrak{p}$	$\Theta(n)\mathfrak{m} + \Theta(1)\mathfrak{p}$
$\mathbf{NP}$-complete lang.	CIRC.-SAT	CIRC.-SAT	SET-PARTITION, SUBSET-SUM, DECISION-KNAPSACK	CIRC.-SAT
Modular	yes	yes	yes	no

We summarize our result in the following informal theorem.

Theorem 2. *Our modular NIZK arguments for* $\mathbf{NP}$*-complete languages are perfectly complete and perfectly zero-knowledge. Under the PSDL assumption, PKE*

Table 3.2: Rewriting the comparison in Table 3.1 using the parameter n only, and setting $N = r_3^{-1}(n) = n^{1+o(1)}$. The best efficiency among the modular NIZK arguments are highlighted in bold.

	Groth	Lipmaa	Our result	Gennaro et al.
$\lvert$CRS$\rvert$	$\Theta(n^2)$	$n^{1+o(1)}$	$n^{1+o(1)}$	$\Theta(n)$
Comm.	$\boldsymbol{\Theta}(\mathbf{1})$	$\boldsymbol{\Theta}(\mathbf{1})$	$\boldsymbol{\Theta}(\mathbf{1})$	$\Theta(1)$
Prover	$\Theta(n^2)\mathfrak{e}$	$(n^{1+o(1)})\mathfrak{e}$	$(\mathbf{n^{1+o(1)}})\mathfrak{m}$	$\Theta(n)\mathfrak{m} + \Theta(n)\mathfrak{e}$
Verifier	$\boldsymbol{\Theta}(\mathbf{n})\mathfrak{m} + \boldsymbol{\Theta}(\mathbf{1})\mathfrak{p}$	$\Theta(n)\mathfrak{e}$	$\boldsymbol{\Theta}(\mathbf{n})\mathfrak{m} + \boldsymbol{\Theta}(\mathbf{1})\mathfrak{p}$	$\Theta(n)\mathfrak{m} + \Theta(1)\mathfrak{p}$
$\mathbf{NP}$-complete lang.	CIRC.-SAT	CIRC.-SAT	SET-PARTITION, SUBSET-SUM, DECISION-KNAPSACK	CIRC.-SAT
Modular	yes	yes	yes	no

assumption in $\mathbb{G}_1$, and PKE assumption in $\mathbb{G}_2$, they are adaptively computationally sound. Moreover, we achieve better efficiency compared to prior modular NIZK arguments for $\mathbf{NP}$-complete languages.

CHAPTER 4

NIZK ARGUMENTS FOR SET OPERATIONS

4.1 Motivation

In many situations, we would like to prove that we meet some requirements, but inadvertently reveal much more in the process. For instance, if Sybil wants to prove she is old enough to go to a casino, she must provide some personal documentation, which also contains additional information such as full name or exact age. Meanwhile, if Dora wants to prove she has enough money to cover travel costs in a foreign country, she will find it hard to do so without revealing how much she actually has in the bank. In both cases, it would be preferable to prove these facts without revealing anything else. As mentioned in the previous chapter, a proof that some integer is in a certain range is known as a range proof.

On a related scenario, consider the voting process where you can vote for multiple candidates. In this instance, you need to prove that you voted for each candidate at most once, without revealing anything about who you did or didn't vote for. This is related to the slightly harder situation of a set membership proof [20], where a prover must show it has chosen a value which is in the public set of allowable values.

Any set membership proof can be adapted to a range proof, but not necessarily the other way around. Moreover, it would be more advantageous to not only be able to prove set membership, but other set operations as well. In this case, a prover must prove that he commits to sets or elements that satisfy a certain relation. However, this must not reveal any information about the individual sets themselves. This type of proof is called a zero-knowledge proof for set operations.

In some cases, it is perfectly fine for a proof for set relations to be interactive. But if we consider the voting scenario, voters might not always be available to interactively prove they have acted honestly. Hence it would be more practical for this zero-knowledge proof for set operations to be non-interactive and transferable.

4.2 Previous Work

Kate, Zaverucha and Goldberg [63] proposed an interactive zero-knowledge proof that a publicly known element x belongs to a committed set S, which was later refined by Henry and Goldberg [61]. These constructions are already very efficient, with linear (in set size $k = |S|$) prover's computation, constant communication and constant verification. However, making this construction non-interactive would need either a loss of efficiency or use of a random oracle.

Kissner and Song [65] constructed an interactive zero-knowledge proof for set union and set intersection which worked with multisets. However, the proof size, prover's computation and verifier's computation were all $O(k)$. Jarecki and Liu [62] improved this to get a zero-knowledge proof for set union with proof size, prover's computation and verifier's computation all $\Theta(k)$. These constructions can be made non-interactive but still efficient using the Fiat-Shamir heuristic [45] (in the RO model) or Ciampi et al. transform [29] (in the RO and CRS model).

4.3 Problem Statement

Is it possible to construct a NIZK argument for set operations in the CRS model that is as efficient as the best NIZK arguments in the RO model? Moreover, is it possible to do so for more set operations than previous work?

4.4 Our Solution

We first construct a pairwise multiset sum equality test (PMSET), which is a NIZK argument that shows that committed multisets $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4$ satisfy the multiset relation $\mathcal{A}_1 \uplus \mathcal{A}_2 = \mathcal{A}_3 \uplus \mathcal{A}_4$. We then use this PMSET argument as a black box to create NIZK arguments for various set operations. A full description of our construction is given in the paper [42], which is a joint work with Lipmaa and Zhang.

4.4.1 Committing to a Multiset

To commit (in $\mathbb{G}_1$) to a multiset $\mathcal{A}$ of cardinality at most k, we first encode $\mathcal{A}$ as a polynomial $f_{\mathcal{A}}(X) = \prod_{a \in \mathcal{A}} (X - a) \in \mathbb{Z}_p[X]$, taking multiplicity into account.

For some values $\bar{\boldsymbol{a}} \in \mathbb{Z}_p^{k+1}$, this can be rewritten as

$$f_{\mathcal{A}}(X) = \sum_{i=0}^{k} \bar{a}_i X^i \enspace .$$

We then commit to $\bar{\boldsymbol{a}}$ using the $(P_i(X))_{i=0}^{k}$-PolyCommit scheme of Section 2.11.1 with $P_i(X) = X^i$ and $P_0(X) = X^v = X^{k+1}$. So we get the commitment

$$\mathsf{Com}_1(\mathcal{A}; r) = g_1^{f_{\mathcal{A}}(\chi) + r\chi^v} = g_1^{r\chi^v + \sum_{i=0}^{k} \bar{a}_i \chi^i} \enspace .$$

The commitment scheme is similarly defined in $\mathbb{G}_2$, and is denoted by $\mathsf{Com}_2(\mathcal{A}; r)$. To achieve soundness of the PMSET argument, we need to commit in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

4.4.2 The PMSET Argument

To get the PMSET argument, we use the fact that $\mathcal{A}_1 \uplus \mathcal{A}_2 = \mathcal{A}_3 \uplus \mathcal{A}_4$ iff $f_{\mathcal{A}_1}(X) f_{\mathcal{A}_2}(X) = f_{\mathcal{A}_3}(X) f_{\mathcal{A}_4}(X)$. But if $f_{\mathcal{A}_1}(\chi) f_{\mathcal{A}_2}(\chi) - f_{\mathcal{A}_3}(\chi) f_{\mathcal{A}_4}(\chi) = 0$ for some $\chi \leftarrow_r \mathbb{Z}_p$, then by the Schwartz–Zippel lemma, we have that with overwhelming probability, $f_{\mathcal{A}_1}(X) f_{\mathcal{A}_2}(X) - f_{\mathcal{A}_3}(X) f_{\mathcal{A}_4}(X) = 0$ as a polynomial. Hence it suffices to show that $f_{\mathcal{A}_1}(\chi) f_{\mathcal{A}_2}(\chi) = f_{\mathcal{A}_3}(\chi) f_{\mathcal{A}_4}(\chi)$.

If the multiset commitments were not randomized, then we can just check that

$$\hat{e}(g_1^{f_{\mathcal{A}_1}(\chi)}, g_2^{f_{\mathcal{A}_2}(\chi)}) = \hat{e}(g_1^{f_{\mathcal{A}_3}(\chi)}, g_2^{f_{\mathcal{A}_4}(\chi)}) \enspace .$$

To keep the multisets private, we replace $g_1^{f_{\mathcal{A}_1}(\chi)}$ by $D_1 = \mathsf{Com}_1(\mathcal{A}_1; r_1)$, $g_2^{f_{\mathcal{A}_2}(\chi)}$ by $D_2 = \mathsf{Com}_2(\mathcal{A}_2; r_2)$, etc. We then get an equation of the form

$$\hat{e}(D_1, D_2) = \hat{e}(D_3, D_4) \cdot \hat{e}(g_1, g_2^E) \enspace , \quad \text{where}$$

$$E = (f_{\mathcal{A}_1}(\chi) + r_1\chi^v)(f_{\mathcal{A}_2}(\chi) + r_2\chi^v) - (f_{\mathcal{A}_3}(\chi) + r_3\chi^v)(f_{\mathcal{A}_4}(\chi) + r_4\chi^v)$$

accounts for the use of randomness in the commitments.

However, to achieve soundness we essentially need to commit to each multiset in both $\mathbb{G}_1$ and $\mathbb{G}_2$, resulting in commitments C_j, D_j for each $\mathcal{A}_j$, $j \in [1\,..\,4]$. This will make the final PMSET argument more complicated, e.g., we need to also prove that C_j and D_j commit to the same multiset, which requires an extra secret variable and extra knowledge components for $j \in [1\,..\,4]$.

The resulting PMSET argument is perfectly complete, perfectly zero knowledge, computationally sound and an argument of knowledge under the PSDL and PKE assumptions. We achieve linear (in the number of set elements) CRS and prover's complexity, and constant communication and verifier's complexity.

4.4.3 Applications of PMSET

The PMSET argument can be used as a black box to construct various set operations. For instance, to prove $A \subseteq B$, we can produce a multiset C such that $A \uplus C = B \uplus \emptyset$. To prove a multiset A is a set, we first publish a public set U which contains all the allowed elements in the multiset. Then we show that $A \subseteq U$.

Set intersection and set union are less obvious to construct. We do this simultaneously, i.e., we prove four committed (multi)sets A, B, C, D satisfy both $C = A \cap B$ and $D = A \cup B$ iff all the following properties hold:

- $A \uplus B = C \uplus D$,
- $C \subseteq A$ and $C \subseteq B$, and
- A, B, D are sets.

Using a similar trick, we can also get set difference. However, we did not achieve multiset intersection, multiset union, or multiset difference.

Additionally, the PMSET argument can be used to construct cryptographic accumulators [10], where given a committed set S and a public integer k, one has to provide a short proof of either $k \in S$ or $k \notin S$. Using PMSET, an accumulator can be defined as follows.

- To prove $k \in S$, create a PMSET argument proving $\{k\} \cup S' = S$ for some committed multiset S', and a PMSET argument proving S is a set.
- To prove $k \notin S$, create a PMSET argument proving $\{k\} \cup S' = U \setminus S$ for some committed multiset S', and a PMSET argument proving S' is a set.

Furthermore, we can implement a dynamic accumulator [22], where one can dynamically add or delete from an accumulated set. Let S be the set to be accumulated. Using PMSET, we can define the dynamic add and dynamic delete operations as follows.

- Dynamic add: show that $k \notin S$, then with S' defined as the set such that $\{k\} \cup S' = U \setminus S$, use the commitment to $U \setminus S'$ as the accumulator for $S \cup \{k\}$.
- Dynamic delete: show that $k \in S$, then with S' defined as the set such that $\{k\} \cup S' = S$, use the commitment to S' as the accumulator for $S \setminus \{k\}$.

Table 4.1: A comparison of our NIZK arguments for set operations [42] with the Kate et al. [63], Kissner-Song [65] and Jarecki-Liu [62] arguments in the RO model.

	Kate et al.	Kissner-Song	Jarecki-Liu	Our result
$\vert\mathrm{CRS}\vert$	$\Theta(k)$	-	$\mathbf{\Theta(1)}$	$\Theta(k)$
Comm.	$\mathbf{\Theta(1)}$	$O(k)$	$\Theta(k)$	$\mathbf{\Theta(1)}$
Prover	$\Theta(k)$	$\mathbf{O(k)}$	$\Theta(k)$	$\Theta(k)$
Verifier	$\mathbf{\Theta(1)}$	$O(k)$	$\Theta(k)$	$\mathbf{\Theta(1)}$
RO	yes	yes	yes	**no**
ZK-sets	yes			yes
Committed subset				yes
Set intersection		yes	yes	yes
Set union		yes		yes
Set difference				yes
Accumulator				yes

4.5 Comparison with Previous Work

Our NIZK argument is as efficient as the zero-knowledge sets construction of Kate et al. [63] in all parameters. Our NIZK set intersection argument is slightly less efficient than Kissner-Song [65] in prover's computation, and less efficient than Jarecki-Liu [62] in CRS length, but is more efficient in both communication and verifier's computation. Hence, our NIZK argument for set operations is at least as efficient as comparable constructions in the RO model. Moreover, we provide a richer library of set operations, and can use our PMSET argument to construct set membership, range argument, and accumulator. A more precise comparison can be seen in Table 4.1.

We summarize our result in the following informal theorem.

Theorem 3. *Our modular NIZK arguments for set relations are perfectly complete and perfectly zero-knowledge. Under the PSDL assumption, PKE assumption in $\mathbb{G}_1$, and PKE assumption in $\mathbb{G}_2$, they are adaptively computationally sound. Moreover, we achieve efficiency comparable to prior NIZK arguments for set relations in the RO model, and for a richer library of set relations.*

CHAPTER 5

NIZK ARGUMENT FOR SHUFFLE

5.1 Motivation

In traditional paper voting, a voter goes to a voting centre, where he takes a ballot paper and votes in a ballot booth outside prying eyes. He then proceeds to seal the ballot paper (either folds it or puts it in a prescribed envelope), which he finally puts into a locked ballot box. When the voting period ends, the voting committee will be able to unlock the ballot box and tally the votes publicly.

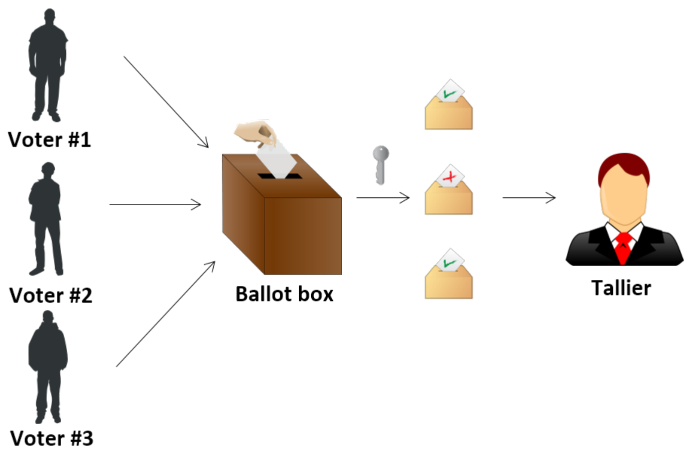

Figure 5.1: Traditional paper voting using a ballot box. To protect voters' privacy, the box must be shaken before unlocking the ballot box and tallying the votes.

The simple scenario above has an obvious weakness in terms of voter's privacy. An observer who has seen the order of voters' ballots going in and out of the ballot box can get a very good guess on who voted for what. This can be remedied by having one or more people shaking the ballot box before it is unlocked for the tallying process. If at least one person does this diligently, then this process will mostly remove the relationship between voters and their votes. However, it is not clear how to check whether or not the process of shaking the ballot was done diligently.

In electronic voting, we also want similar properties. A voter must be able to send their votes to a vote collecting server and have the vote stay secret until the tallying process, but the tallier (or anyone else, for that matter) must not be able to infer what any voter voted for from the set of opened votes. In cryptography, we know how to hide messages using public key encryption schemes that are IND-CPA secure. However, if the decrypter receives the voter's ciphertexts directly from the voting server, the voters' privacy is again breached by a curious decrypter. Hence we also want to remove the association between voters and the ciphertexts that reach the decrypter. This can be done by the use of mix-servers, combined into a mix-network (or mix-net for short), which will individually permute and re-randomize the ciphertexts, before sending it to the next mix-server in the mix-net. (Another option is to have a decryption mix-net [27] where each mix-server shuffles and decrypts one layer of the ciphertext.)

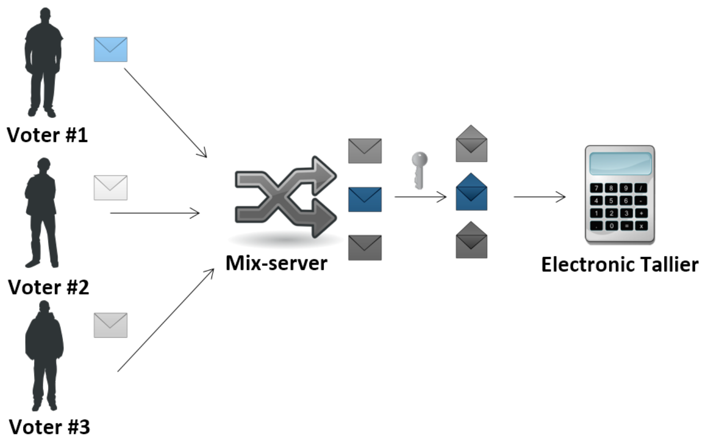

Figure 5.2: Electronic voting analogy. To protect voters' privacy, the votes must be shuffled before decrypting and tallying the votes.

To enable re-randomization of ciphertexts, we must use an encryption scheme that is homomorphic, such as ElGamal. A bigger problem is how to prove that each mix-server did its shuffle correctly, without revealing the permutation or randomness values it used. The proof must be relatively short, transferable, and verifiable at any time, even in a much later audit process when the mix-servers have gone offline. Each mix-server thus has to provide a non-interactive zero-knowledge shuffle argument.

Assume that there are M mix-servers between the vote collecting server and the decrypter. In some cases, the voting server is considered to be the the 0th (non-mixing) mix-server, and the decrypter is considered the $(M + 1)$th mix-server. To ensure security against malicious mix-servers, the kth mix-server checks the shuffle argument of all the previous mix-servers, before creating its own shuffle argument to send to the $(k + 1)$th mix-server. To get robustness, if the shuffle argument of there first i mix-servers is valid, but the $(i + 1)$th is not, then the $(i + 1)$th mix-server and all subsequent ones are ignored, and the kth mix-server creates its shuffle argument using the output of the ith mix-server. This means that the verifier's computation is a bottleneck, especially for the mix-servers close to the decrypter.

We note that before the final tallying, the decrypter would also need to prove that it decrypts the final set of ciphertexts correctly. This decrypter is usually implemented as a multi-party computation. However, we will omit this part in our discussion.

5.2 Previous Work

There have been several results in interactive zero-knowledge shuffle arguments, which can be made non-interactive in the RO model using the Fiat-Shamir heuristic. Of these, Bayer and Groth [6] constructed the shuffle argument with most efficient communication, and Furukawa [47] constructed the shuffle argument with least number of rounds. The most computationally efficient interactive zero-knowledge shuffle argument that we know of is by Groth [55]. It requires 7 rounds of interaction, while still needing a CRS. In the CRS model (and without using random oracles), there were two known NIZK shuffle arguments before our work, namely by Groth and Lu [58], and Lipmaa and Zhang [71].

The Groth-Lu shuffle argument, constructed using Groth-Sahai proofs, is culpably sound under one well-known and two newly defined computational assumptions. Culpable soundness in this case means that if an adversary can create an accepting argument for an incorrect statement (i.e. the output set of ciphertexts is not a shuffle of the input set of ciphertexts), then this adversary can work together with a party that knows the secret key to break one of the stated computational assumptions. However, these computational assumptions are defined in such a way that knowledge of the secret key does not help in breaking them. Culpable soundness is a weaker notion than adaptive soundness (where collaboration with a party that knows the secret key is not necessary), but is still acceptable in the case of electronic voting, since one can assume that some coalition of parties knows the secret key. The Groth-Lu shuffle argument is significantly less efficient than the Groth [55] RO model shuffle argument.

The Lipmaa-Zhang shuffle argument is sound if there exists an extractor that can access the adversary's randomness values. Hence if an adversary can create an accepting argument, by using the extractor it can be proven that the output set of ciphertexts is indeed a shuffle of the input set of ciphertexts, except for a negligible probability. We refer to this notion of soundness as white-box soundness [39], which is a weaker notion than culpable soundness. Moreover, Lipmaa and Zhang use the knowledge BBS cryptosystem (the standard BBS cryptosystem which they modified to have a knowledge component) which is lifted (encrypts integers rather than group elements), and to achieve soundness they additionally need to prove that the plaintexts are small. The Lipmaa-Zhang shuffle argument is more efficient than the Groth-Lu shuffle argument, but still significantly less efficient than the RO-based Groth [55] shuffle argument.

5.3 Problem Statement

Is it possible to construct a NIZK shuffle argument in the CRS model that is much closer in efficiency to the best NIZK shuffle arguments in the RO model?

5.4 Our First Solution

Let n be the number of ciphertexts, which we encrypt with regular (non-lifted) ElGamal. We construct a NIZK shuffle argument by committing to an $n \times n$ permutation matrix $\boldsymbol{\Psi}$, and proving that it is used to permute the original set of ciphertexts. We then extend this to work even if the permuted set of ciphertexts is re-randomized. Similar to the Groth-Lu shuffle, we relax our security requirements, where instead of regular soundness we use culpable soundness. A full description of our construction is given in the paper [39], which is a joint work with Lipmaa.

5.4.1 Committing to a Matrix

We commit to matrix $M_{n\times n}$ by committing to each of the n rows separately. Moreover, we commit to a row $\boldsymbol{m} = (m_1, \cdots, m_n)$ by using the $(P_i(X))_{i=0}^n$-$\mathsf{PolyCommit}$ scheme of Section 2.11.1 with a knowledge component, using well-chosen polynomials $(P_i(X))_{i=0}^n$ and knowledge secret γ, as follows:

$$\mathsf{Com}(\mathsf{ck}; \boldsymbol{m}; r) = (g_1, g_2^\gamma)^{rP_0(\chi)+\sum_{i=1}^n m_i P_i(\chi)} \quad .$$

5.4.2 Permutation Matrix Argument

The prover must prove that it commits to a permutation matrix. Due to Corollary 2, it is sufficient to prove that the first $n-1$ rows $\boldsymbol{\Psi}_i$ of $\boldsymbol{\Psi}$ (transposed) are unit vectors, and that $\boldsymbol{\Psi}_n = \mathbf{1}_n - \sum_{i=1}^{n} \boldsymbol{\Psi}_i$ is also a unit vector. Hence we only need to commit to the first $n-1$ rows of the permutation matrix.

Due to Lemma 1, we can show that $\boldsymbol{a}$ is a unit vector iff $a_i \in \{0,1\}$ for $i \in [1\,..\,n]$ (i.e., $\boldsymbol{a}$ is Boolean), and $\sum_{i=1}^{n} a_i = 1$. Define $\boldsymbol{V} = \left(\begin{smallmatrix} 2 \cdot I_{n \times n} \\ \mathbf{1}_n^T \end{smallmatrix}\right)$, and $\boldsymbol{b} = \left(\begin{smallmatrix} \mathbf{0}_n \\ 1 \end{smallmatrix}\right)$.

For a vector $\boldsymbol{a} \in \mathbb{Z}_p^n$, we have that

$$V\boldsymbol{a} + \boldsymbol{b} = \begin{pmatrix} 2a_1 \\ 2a_2 \\ \vdots \\ 2a_n \\ \sum_{i=1}^{n} a_i \end{pmatrix} .$$

Hence (assuming $n < p - 1$) the $n+1$ conditions above hold iff

$$\begin{aligned} V\boldsymbol{a} + \boldsymbol{b} &\in \{0,2\}^{n+1} \\ \iff (V\boldsymbol{a} + \boldsymbol{b}) \circ (V\boldsymbol{a} + \boldsymbol{b} - 2 \cdot \mathbf{1}_{n+1}) &= \mathbf{0}_{n+1} \\ \iff (V\boldsymbol{a} + \boldsymbol{b} - \mathbf{1}_{n+1}) \circ (V\boldsymbol{a} + \boldsymbol{b} - \mathbf{1}_{n+1}) &= \mathbf{1}_{n+1} \ . \end{aligned}$$

Hence, $\boldsymbol{a}$ is a unit vector iff

$$(\boldsymbol{V}\boldsymbol{a} + \boldsymbol{b} - \mathbf{1}_{n+1}) \circ (\boldsymbol{V}\boldsymbol{a} + \boldsymbol{b} - \mathbf{1}_{n+1}) = \mathbf{1}_{n+1} \ . \tag{5.1}$$

We use the square span programs (SSP, [30]) approach that makes use of Equation 5.1 to get a very efficient unit vector argument. To achieve this we choose $P_i(X) = y_i(X)$ to be polynomials that interpolate the i-th column of $\boldsymbol{V}$ for $i \in [1\,..\,n]$, and $y_0(X)$ to be the polynomial that interpolates $\boldsymbol{b} - \mathbf{1}_{n+1}$.

5.4.3 Shuffle Argument

For $a = (a_1, a_2) \in \mathbb{G}_1^2$ and $b \in \mathbb{G}_2$, we define the double pairing $\hat{E}$ as $\hat{E}(a, b) = (\hat{e}(a_1, b), \hat{e}(a_2, b))$.

Let $\boldsymbol{z}$ be the set of original ElGamal ciphertexts, and let $\boldsymbol{z'}$ be the set of shuffled ciphertexts. Let $\mathsf{pk} = (g_1, h)$ be the ElGamal public key. For $i \in [1\,..\,n]$, let $z_i = \mathsf{Enc}_{\mathsf{pk}}(m_i; u_i) = (g_1^{u_i}, m_i \cdot h^{u_i})$ and $z'_i = z_{\psi(i)} \cdot \mathsf{Enc}_{\mathsf{pk}}(m_i; t_i) =$

$(g_1^{u_{\psi(i)}+t_i}, m_{\psi(i)} \cdot h^{u_{\psi(i)}+t_i})$. If the ciphertexts $\boldsymbol{z'}$ were permuted but not re-randomized (i.e. $t_i = 0$ and $z'_i = z_{\psi(i)}$), we could verify that $z'_i = z_{\psi(i)}$ simply by checking that

$$\prod_{i=1}^{n} \hat{E}(z'_i, g_2^{\gamma P_i(\chi)}) = \prod_{i=1}^{n} \hat{E}(z_i, g_2^{\gamma P_{\psi^{-1}(i)}(\chi)}) \ .$$

Note that if $\gamma \neq 0$, this holds iff the equation

$$\sum_{i=1}^{n} P_i(\chi) u_{\psi(i)} = \sum_{i=1}^{n} P_{\psi^{-1}(i)}(\chi) u_i$$

also holds.

However, just comparing $\boldsymbol{z}$ and $\boldsymbol{z'}$ will clearly reveal the permutation ψ. Moreover, to do this the verifier needs to know the values $g_2^{\gamma P_{\psi^{-1}(i)}(\chi)}$. Also, as we will discuss below, it is not clear if this is sufficient to get soundness.

To ensure privacy, $\boldsymbol{z'}$ needs to be both permuted and randomized (i.e. $z'_i = z_{\psi(i)} \cdot \mathsf{Enc}_{\mathsf{pk}}(1; t_i)$ for some randomness value t_i), and we need to replace $g_2^{P_{\psi^{-1}(i)}(\chi)}$ with the value c_{i2}^{γ}, where

$$(c_{i1}, c_{i2}^{\gamma}) = (g_1, g_2^{\gamma})^{r_i P_0(\chi) + P_{\psi^{-1}(i)}(\chi)}$$

is a commitment of $\boldsymbol{e}_{\psi^{-1}(i)}$, the $\psi^{-1}(i)$-th unit vector. This makes the verification equation become slightly more complicated. In particular, we need an extra error term $\mathcal{E}$ such that the check becomes

$$\prod_{i=1}^{n} \hat{E}(z'_i, g_2^{\gamma P_i(\chi)}) = \prod_{i=1}^{n} \hat{E}(z_i, c_{i2}^{\gamma}) \cdot \mathcal{E}. \tag{5.2}$$

Note that $\mathcal{E}$ depends on the CRS and the randomness values used in re-randomizing the ciphertexts and committing the permutation matrix.

Same-message argument

Unfortunately, the verification equation Equation 5.2 is still not enough. This is because an adversary can still make use of the CRS values and the fact that $P_i(X)P_j(X) = P_j(X)P_i(X)$ for all $i, j \in [1 \mathinner{..} n]$ to create $\boldsymbol{z'}$ which is not a shuffle of $\boldsymbol{z}$, but still satisfies Equation 5.2.

To fix this, we commit to the same permutation matrix, but with a different (well-chosen) set of polynomials $(\hat{P}_i(X))_{i=0}^{n}$. We then use the resulting commitments $(\hat{c}_{i1}, \hat{c}_{i2}^{\gamma})$ to get a second verification equation

$$\prod_{i=1}^{n} \hat{E}(z'_i, g_2^{\gamma \hat{P}_i(\chi)}) = \prod_{i=1}^{n} \hat{E}(z_i, \hat{c}_{i2}^{\gamma}) \cdot \hat{\mathcal{E}} \ , \tag{5.3}$$

for some value $\hat{\mathcal{E}}$. However, we then need to prove that $(c_{i1}, c_{i2}^{\gamma})$ and $(\hat{c}_{i1}, \hat{c}_{i2}^{\gamma})$ commit to the same vector. We call this a same-message argument. The idea of committing the permutation matrix with a different set of polynomials was also used by Groth and Lu [58], but in their case the construction is trivial, since they used independent random variables $(X_i)_{i=0}^n$ instead of polynomials $(P_i(X))_{i=0}^n$, and $(X_i)_{i=0}^n$ instead of $(\hat{P}_i(X))_{i=0}^n$. Our same-message argument is more involved, and is again constructed using square span programs.

PSP assumption

Recall that a prover must provide two sets of ciphertexts such that Equation 5.2 and Equation 5.3 hold, where $(c_{i1}, c_{i2}^{\gamma})_{i=1}^n$ and $(\hat{c}_{i1}, \hat{c}_{i2}^{\gamma})_{i=1}^n$ are commitments to the same permutation matrix, but using different commitment keys corresponding to different sets of polynomials $(P_i(X))_{i=0}^n$ and $(\hat{P}_i(X))_{i=0}^n$. We claim that a malicious prover who sends a shuffle argument which is accepted by the verifier will succeed with negligible probability. This claim is simplified into an assumption that is easier to analyze, which we call the Power Simultaneous Product (PSP) assumption.

Definition 15. Let $(P_i(X))_{i=0}^n$ and $(\hat{P}_i(X))_{i=0}^n$ be tuples of polynomials. Let $d = \max\deg P_i(X)$. The $((P_i(X))_{i=0}^n, (\hat{P}_i(X))_{i=0}^n)$-PSP assumption states that any NUPPT adversary $\mathcal{A}$, given values $((g_1, g_2)^{\chi^i})_{i=0}^d, ((g_1, g_2)^{\hat{P}_i(\chi)})_{i=0}^n)$ where $\chi \leftarrow_r \mathbb{Z}_p$, has negligible probability of producing values $(\boldsymbol{s} \neq \mathbf{1}_n, t, \hat{t})$ such that

$$t^{P_0(\chi)} \cdot \prod_{i=1}^n s_i^{P_i(\chi)} = \hat{t}^{\hat{P}_0(\chi)} \cdot \prod_{i=1}^n s_i^{\hat{P}_i(\chi)} = 1 \ .$$

This assumption is related to the SP assumption of Groth and Lu [58], and can be seen as a generalization of the matrix computational assumptions of Morillo, Ràfols and Villar [74]. If we take a discrete log of the equations, the adversary aims to output values $(\boldsymbol{s'} \neq \mathbf{0}_n, t', \hat{t}')$ such that the polynomials

$$\begin{aligned} S(X) &= t' P_0(X) + \sum_{i=1}^n P_i(X) \\ \hat{S}(X) &= t' \hat{P}_0(X) + \sum_{i=1}^n \hat{P}_i(X) \end{aligned}$$

satisfy $S(\chi) = \hat{S}(\chi) = 0$. I We prove that if the polynomials $(P_i(X))_{i=0}^n$ and $(\hat{P}_i(X))_{i=0}^n$ are well chosen, the PSP assumption holds in the GBGM [39].

Culpable soundness

We prove our shuffle argument is culpably sound with respect to the following relation:

$$\mathcal{R}^{\mathsf{guilt}}_{sh,n} = \left\{ \begin{array}{l} (\mathsf{gk}, (\mathsf{pk}, (z_i)_{i=1}^{n}, (z_i')_{i=1}^{n}), \mathsf{sk}) : \\ \mathsf{gk} \in \mathsf{BP}(1^\kappa, n) \wedge (\mathsf{pk}, \mathsf{sk}) \in \mathsf{genpkc}(\mathsf{gk}) \wedge \\ (\forall \psi \in S_n : \exists i : \mathsf{Dec}_{\mathsf{sk}}(z_i') \neq \mathsf{Dec}_{\mathsf{sk}}(z_{\psi(i)})) \end{array} \right\} .$$

Essentially, this means that if a malicious adversary cheats, then since he knows the secret key, he knows the plaintexts he uses to cheat, and they are not shuffles of each other.

Using the methods above, we get a perfectly complete, culpably sound, and perfectly zero-knowledge shuffle argument under the XDH, PCDH, and TSDH computational assumptions, the PKE knowledge assumption, and the PSP assumption.

5.5 Our Second Solution

Again, let $\boldsymbol{z}$ be the set of original ciphertexts, and let $\boldsymbol{z'}$ be the set of shuffled ciphertexts. In the previous solution, we see that the verifier's computation is dominated by checking both Equation 5.2 and Equation 5.3 related to the two sets of ciphertexts. This was necessary, since only checking one such solution did not result in a sound shuffle argument, and additional verification steps were needed to get soundness.

In our second solution, we construct a NIZK shuffle argument that improves the previous construction by only needing one such verification equation. This means that we no longer need to commit to the same permutation matrix twice, and hence we no longer need the same-message argument. Instead, using the technique used in the Lipmaa-Zhang shuffle argument, we require the ciphertext to essentially be encrypted in both $\mathbb{G}_1$ and $\mathbb{G}_2$, and prove that they encrypt the same message, which could only be computed from a specific small subset of CRS values. This rules out the attack against only one verification equation of the form Equation 5.2 that was possible in the first solution, and in fact results in a sound shuffle argument. However, as will be evident in Section 5.5.3, in this setting ElGamal is no longer IND-CPA secure, hence we encrypt the ciphertexts using lLin instead.

We prove the soundness (and not culpable soundness) of the whole shuffle argument in the GBGM. Essentially, we assume a single adversary that provides a shuffle argument that is accepted by an honest verifier. In the GBGM, we require that the adversary knows how he gets these values as a product or pairing of the

CRS values. The CRS consists of values of type $h = g_i^{H(\chi)}$, for known (multivariate) polynomials $H(\boldsymbol{X})$, where χ is a tuple of integers, each taken uniformly random from $\mathbb{Z}_p$. Taking a discrete log, this means that verification equations such as $\hat{e}(A, B) = \hat{e}(g_1, C)$ are equivalent to polynomial equations such as $V(\chi) = 0$ for $V(\boldsymbol{X}) = A(\boldsymbol{X})B(\boldsymbol{X}) - C(\boldsymbol{X})$. By the Schwartz–Zippel lemma, this means that the verification equations hold as a system of polynomial equations $V(\boldsymbol{X}) = 0$. However, if $V(\boldsymbol{X}) = 0$, then every monomial of this polynomial has to have coefficient zero, leading to a bigger system of polynomial equations. Using automated tools such as Mathematica and wxMaxima, we then find the Gröbner basis for the polynomial equations, and from solving this (also using automated tools) derive soundness.

It can be argued that working in the GBGM is more reasonable than using knowledge assumptions as in the Lipmaa-Zhang shuffle argument and our first construction. Although soundness in the GBGM does not rule out existence of adversarial attacks, it does imply that any possible attack is not generic and must use the specific structure of the chosen bilinear group. In comparison, it is known that knowledge assumptions do not hold if the auxiliary input is not well chosen [11], and hence they must be very carefully formulated. Moreover, knowledge assumptions must at least be proven to hold in the GBGM, so it makes sense to just work directly in the GBGM. As we have mentioned above, the use of GBGM enables computerized analysis for many of the tedious parts of the soundness proof.

A full description of our construction is given in the paper [40], which is a joint work with Lipmaa and Zając.

5.5.1 Committing to a Matrix

As in the first solution, commit to matrix $M_{n\times n}$ by committing to each of the n rows separately. Similarly, we commit to a row $\boldsymbol{m} = (m_1, \cdots, m_n)$ by using the $(P_i(X))_{i=0}^n$-PolyCommit scheme of Section 2.11.1 using well-chosen polynomials $(P_i(X))_{i=1}^n \cup \{\varrho\}$ as follows:

$$\mathsf{Com}(\mathsf{ck}; \boldsymbol{m}; r) = (g_1, g_2)^{r\varrho(\chi) + \sum_{i=1}^n m_i P_i(\chi)} \enspace .$$

Note that we do not need a knowledge value γ here. Moreover, the verifier does not need to check the consistency of the first and second component of the commitment. This makes the constructed arguments more efficient.

5.5.2 Permutation Matrix Argument

The prover must prove that it commits to a permutation matrix. Due to Corollary 2, it is sufficient to prove that the first $n-1$ rows $\boldsymbol{\Psi}_i$ of $\boldsymbol{\Psi}$ (transposed) are 1-sparse vectors, and that $\boldsymbol{\Psi}_n = \boldsymbol{1}_n - \sum_{i=1}^n \boldsymbol{\Psi}_i$ is also a 1-sparse vector. This method was also used in the Lipmang-Zhang shuffle argument [71]. Similar to the first solution, we only need to commit to the first $n - 1$ rows of the permutation matrix.

As in the first solution, we use SSP to prove that Equation 5.1 holds for $\boldsymbol{a}' = k \cdot \boldsymbol{a}$, for some constant k. This will show that $k \cdot \boldsymbol{a}$ is a unit vector, i.e., $\boldsymbol{a}$ is 1-sparse. Recall that $P_i(X)$ are polynomials that interpolate the i-th column of $\boldsymbol{V}$ for $i \in [1 \mathinner{..} n]$, and $P_0(X)$ interpolates $\boldsymbol{b} - \mathbf{1}_{n+1}$ for $i = 0$.

Using techniques introduced in a recent work of Groth [57], this can be verified using only n pairing equations. We prove the resulting permutation matrix argument is complete, witness-indistinguishable, and computationally sound in the GBGM.

5.5.3 Shuffle Argument

For $a = (a_1, a_2, a_3) \in \mathbb{G}_1^3$ and $b \in \mathbb{G}_2$, we define the triple pairing $\hat{E}$ as $\hat{E}(a, b) = (\hat{e}(a_1, b), \hat{e}(a_2, b), \hat{e}(a_2, b))$.

Let $\boldsymbol{z}$ be the set of original ILin ciphertexts, and let $\boldsymbol{z'}$ be the set of shuffled ciphertexts. We recall the observation in the first solution that if $\boldsymbol{z'}$ were permuted but not re-randomized, we can verify that $z'_i = z_{\psi(i)}$ simply by checking that

$$\prod_{i=1}^{n} \hat{E}(z'_i, g_2^{P_i(\chi)}) = \prod_{i=1}^{n} \hat{E}(z_i, g_2^{P_{\psi^{-1}(i)}(\chi)}) \enspace .$$

Moreover, to ensure privacy, $\boldsymbol{z'}$ needs to again be be both permuted and randomized, and $g_2^{P_{\psi^{-1}(i)}(\chi)}$ must be replaced by with c_{i2}, where (c_{i1}, c_{i2}) is a commitment of the unit vector $\boldsymbol{e}_{\psi^{-1}(i)}$. Similar to our first solution, this introduces an extra term $\mathcal{E}$ to the verification equation, such that the check becomes Equation 5.2.

The main difference is that while $\mathcal{E}$ stays independent of the permutation and randomness values used in the permutation matrix argument, it is of a slightly different form compared to the term $\mathcal{E}$ used in our previous solution. We call the resulting check a *consistency argument.*

Validity argument

As in the first solution, an adversary can still make use of the CRS values and the fact that $P_i(X)P_j(X) = P_j(X)P_i(X)$ to create $\boldsymbol{z'}$ which is not a shuffle of $\boldsymbol{z}$, but still satisfies Equation 5.2. To fix this, we require the prover to shuffle the ciphertext in both $\mathbb{G}_1$ and $\mathbb{G}_2$, and use a so-called *validity argument* to prove this is done correctly. This restricts the form that $\mathsf{Dec}(z)$ and $\mathsf{Dec}(z')$ can have, and in the process rules out the mentioned adversarial attack.

We use the same secret key sk in both $\mathbb{G}_1$ and $\mathbb{G}_2$, with public key $(g_1^{\varrho/\beta}, g_1^{\mathsf{sk}\cdot\varrho/\beta})$ for $\mathbb{G}_1$ and (g_2, g_2^{sk}) for $\mathbb{G}_2$, for some random variables ϱ, β. The use of the multiplier ϱ/β in the public key for $\mathbb{G}_1$ is done so that the validity argument generates additional equations in the GBGM proof. The obvious cost of this validity argument is the need to compute two sets of shuffled ciphertexts. Additionally, the prover has to deal with lLin ciphertexts in $\mathbb{G}_i^3$ instead of ElGamal ciphertexts in $\mathbb{G}_1^2$ as in the first solution. Consequently, in this part of the shuffle argument, the prover needs to do slightly more computation than in the first solution.

Note that we cannot use ElGamal as in the first solution. First of all, we would need to use lifted ElGamal to be able to encrypt the same message. Furthermore, we no longer have IND-CPA security. This is, assume that we have a public key $(g_1^{\varrho/\beta}, h_1 = g_1^{\mathsf{sk}\cdot\varrho\cdot\gamma/\beta})$ for $\mathbb{G}_1$ and $(g_2, h_2 = g_2^{\mathsf{sk}})$ for $\mathbb{G}_2$. (The random variable γ takes into account all variations of this construction.) Let $z = (g_1^{r\cdot\varrho/\beta}, g_1^{m\cdot\varrho/\beta} h_1^r) \in \mathbb{G}_1^2$ and $z' = (g_2^r, g_2^m h_2^r) \in \mathbb{G}_2^2$ be encryptions of the same message m in $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Then we can easily check whether or not $m = 0$, since it is equivalent to the pairing equation $\hat{e}(z_1, h_2) = \hat{e}(g_1^{\varrho/\beta}, z'_2)$.

The use of lLin in both $\mathbb{G}_1$ and $\mathbb{G}_2$ means that encryption is lifted, and thus decryption is only efficient for small messages. However, due to the validity argument, the soundness proof in the GBGM takes into account adversaries that encrypt potentially large messages. Hence, we do not need an additional range proof as in [71] to get security of the shuffle argument.

Soundness

It turns out that adding the validity arguments is sufficient to get soundness. Using the methods above, we get a perfectly complete, computationally sound, and perfectly zero-knowledge argument in the GBGM. Note that in the GBGM, we do not get black-box soundness as in the first solution. However, compared to the first solution, we achieve standard soundness instead of culpable soundness, and do not use knowledge assumptions to extract an adversary's input.

Batching

We further optimize the verifier's computation by the use of batching, a technique introduced by Bellare, Garay, and Rabin [8], that has not been used in previous CRS-based shuffle arguments. Essentially, instead of checking k verification equations, the verifier generates k random values and uses them to replace this by a single verification equation. The following lemma states that these two verification checks are equivalent, except for a negligible probability. (See [39] for a proof.)

Lemma 4. *Let $(q_i)_{i\in[1\,..\,k]}$ be values chosen uniformly random from $\mathbb{Z}_p$. Let $(F_i(\chi))_{i\in[1\,..\,k]}$ be a tuple of multivariate polynomials F_i evaluated at some value $\boldsymbol{X} = \chi$. If the equation $\prod_{i=1}^{k} \hat{e}(g_1, g_2)^{F_i(\chi)q_i} = 1$ holds for some values $F_i(\chi)$, then with probability $\geq 1 - 1/p$ the k pairing equations $\hat{e}(g_1, g_2)^{F_i(\chi)} = 1$, $i \in [1\,..\,k]$ also hold.*

This in itself does not improve the verifier's computation time, and in fact requires the verifier to do k extra exponentiations. However, under some circumstances, this enables us to replace pairings with exponentiations, which are less costly. For example, if the k verification equations are of the form $\hat{e}(A_i g_1^{F_i(\chi)}, g_2) = 1$, where $g_1^{F_i(\chi)}$ can be computed from the CRS and the values A_i are given by the prover, then the verifier can simply check $\hat{e}(\prod_{i=1}^{k}(A_i g_1^{F_i(\chi)})^{p_i}, g_2) = 1$, which requires k exponentiations but only 1 pairing. Since exponentiation takes much less time than pairings (in the performance result of [17] described below, a pairing takes approximately the same time as 8 exponentiations in $\mathbb{G}_1$, or 4 exponentiations in $\mathbb{G}_2$), this results in a significant improvement in verifier's computation.

5.6 Comparison with Previous Work

Our first NIZK shuffle argument from Section 5.4 is more efficient than the Groth-Lu and Lipmaa-Zhang shuffle arguments in all parameters except the CRS length. By pre-computing values not related to the ciphertexts, the prover essentially only needs to do two $(n + 1)$-wide multi-exponentiations in the online phase. As we have seen above, we encrypt plaintexts using regular ElGamal, and prove that our NIZK shuffle argument is culpably sound. However, we still need to use a knowledge assumption (PKE) to achieve culpable soundness.

A more precise comparison between our first solution and previous work can be seen in Table 5.1. Note that the CRS length and communication are given in total group elements, the prover's computation is given in total group exponentiations, while the verifier's computation is given in number of pairings. The best in each parameter among the shuffle arguments in the CRS model are given in bold.

Our second NIZK shuffle argument from Section 5.5 is more efficient than our first shuffle argument in all parameters except prover's computation. Moreover, we achieve computational soundness instead of culpable soundness. We prove the soundness of the whole shuffle argument in the GBGM, and in the process remove the need for separate computational and knowledge assumptions. Most importantly, we get a much more efficient argument in terms of verifier's computation, which we argued is the bottleneck of mix-nets.

A more precise comparison between our second solution and previous work can be seen in Table 5.2. To have a better idea of running time, we write the

Table 5.1: A comparison of our first NIZK shuffle argument [39] with the Groth-Lu [58] and Lipmaa-Zhang [71] shuffle arguments in the CRS model, and Groth's shuffle argument in the RO model [55]. The CRS length and communication are given in total group elements, the prover's computation is given in total group exponentiations, while the verifier's computation is given in number of pairings.

	Groth-Lu	Lipmaa-Zhang	Our first result	Groth
$\lvert$CRS$\rvert$	$\mathbf{2n+8}$	$7n+6$	$8n+17$	$n+1$
Comm.	$18n+120$	$12n+11$	$\mathbf{9n+2}$	$480n$ bits
Prover	$54n+246$	$28n+11$	$\mathbf{18n+3}$ ($2n+2$ online)	$8n$
Verifier	$75n+282$	$28n+18$	$\mathbf{18n+6}$ ($8n+4$ online)	$6n$ exp.
Sound.	**culpable**	white-box	**culpably sound**	sound
PKE	**no**	yes	yes	no

efficiency values for each group separately. Note that we take n million clock cycles as a basic work unit, using the performance results of Bos, Costello and Naehrig [17] on a Core i7-3520M CPU, where an exponentiation in $\mathbb{G}_1$, an exponentiation in $\mathbb{G}_2$, an exponentiation in $\mathbb{G}_T$, and a pairing take respectively 0.9, 1.8, 3.1, and 7.0 million clock cycles.

Table 5.2: A comparison of our second NIZK shuffle argument [40] with the Lipmaa-Zhang [71] shuffle argument and our first NIZK shuffle argument [39]. The CRS length and communication is given by a triple denoting the number of elements in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$. The computation is given by the quadruple (e_1, e_2, e_T, p), where e_i is the number of exponentiations in group $\mathbb{G}_i$ and p is the number of pairings, along with a weighted sum, denoted by $\approx u$, that approximates the number of work units required to compute it.

	Lipmaa-Zhang	Our first result	Our second result
|CRS|	$(2n+2, 5n+4, 0)$	$(6n+8, 2n+8, 1)$	$(\mathbf{2n+6}, \mathbf{n+7}, \mathbf{1})$
Comm.	$(8n+6, 4n+5, 0)$	$(7n+2, 2n, 0)$	$(\mathbf{4n+1}, \mathbf{3n+2}, \mathbf{0})$
Prover	$(16n+6, 12n+5, 0, 0) \approx 36$ units	$(14n+3, 4n, 0, 0)$ $\approx$ **19.8 units**	$(9n+2, 9n+3, 0, 0)$ ≈ 24.3 units
Verifier	$(0, 0, 0, 28n+18)$ ≈ 196 units	$(0, 0, 0, 18n+6)$ ≈ 126 units	$(11n+5, 3n+6, 1, 3n+6) \approx$ **36.3 units**
Sound.	**sound**	culpable	**sound**
PKE	yes	yes	**no**

CONCLUSION AND FUTURE WORK

In this thesis we provide three scenarios where transferable non-interactive zero-knowledge (NIZK) arguments are important. In the first scenario, we consider verifiable computation and the need for easily checking that a solution to an $\mathbf{NP}$-complete problem is correct. In the second scenario, we look at the challenge of authorization in real world situations, which involve proving some set relation, and privacy requires us to hide one or more of the sets in the relation. In the third scenario, we consider shuffling ciphertexts before decryption as a way to improve privacy of electronic voting systems, by removing the relationship between voters and their votes.

For these scenarios, we construct NIZK arguments in the common reference string (CRS) model which are more efficient than existing work in the CRS model. These NIZK arguments are arguably comparable in efficiency to the best known NIZK arguments in the random oracle (RO) model.

In the first scenario, we get NIZK arguments for $\mathbf{NP}$ that are more efficient than existing ones in the CRS model and use NP-complete languages that are simpler to check than CIRCUIT-SAT, as described in the first publication. In the second scenario, we get NIZK arguments for set operations that are as efficient than existing ones in the RO model but for a bigger library of set operations, as described in the second publication. In the third scenario, we get NIZK shuffle arguments that are more efficient than existing ones in the CRS model, and are almost as efficient as existing ones in the RO model, as described in the third and fourth publications.

However, in all scenarios, there is still room for improvement. For example, in the first three publications, the solutions involve the use of knowledge assumptions, in addition to computational assumptions, as shown in Table 5.3. These knowledge assumptions are not very well studied, and introduce some efficiency loss in the form of computing and verifying knowledge components in the resulting NIZK arguments. For example, in Section 3.4.1, a prover not only has to compute a commitment to a vector $\boldsymbol{a}$ as $C = g_1^{r\sigma^v + \sum_{i=1}^n a_i \sigma^{\lambda_i}}$, but also has to provide a knowledge component $\tilde{C} = (g_1^\gamma)^{r\sigma^v + \sum_{i=1}^n a_i \sigma^{\lambda_i}}$, while the verifier has to check that $\hat{e}(C, g_2^\gamma) = \hat{e}(\tilde{C}, g_2)$. How to improve the proposed solutions in general, and remove knowledge assumptions in particular, is an interesting topic for future work.

Table 5.3: The hardness assumptions used in the published work introduced in this thesis.

Publication	Computational assumptions	Knowledge assumptions
Fauzi, Lipmaa, and Zhang (2013) [41]	PSDL	PKE
Fauzi, Lipmaa, and Zhang (2014) [42]	PSDL	PKE
Fauzi and Lipmaa (2016) [39]	XDH, TSDH, PCDH, PSP	PKE
Fauzi, Lipmaa, and Zając (2016) [40]	-	-

There have been subsequent work that have in fact resulted in such improvements. Lipmaa constructed a significantly improved permutation argument to get a more efficient modular NIZK argument for $\mathbf{NP}$ than our first publication in [68] and also constructed more efficient shift and product arguments in [69], while Groth [57] constructed a non-modular NIZK argument for $\mathbf{NP}$ which is more efficient than existing work and is secure in the generic bilinear group model. Our fourth publication improved the shuffle argument of our third publication and is also secure in the generic bilinear group model. We are not aware of similar improvements over our second publication and our very recent fourth publication.

Bibliography

[1] Abe, M., Fehr, S.: Perfect NIZK with Adaptive Soundness. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 118–136. Springer Berlin Heidelberg, Amsterdam, The Netherlands (Feb 21–24, 2007)

[2] Ateniese, G., Camenisch, J., Hohenberger, S., de Medeiros, B.: Practical Group Signatures without Random Oracles. Tech. Rep. 2005/385, IACR (2005), available at `http://eprint.iacr.org/2005/385/`

[3] Back, A., et al.: Hashcash - A Denial of Service Counter-Measure (2002), available at `http://www.hashcash.org/papers/hashcash.pdf`

[4] Barak, B., Canetti, R., Nielsen, J.B., Pass, R.: Universally Composable Protocols with Relaxed Set-Up Assumptions. In: FOCS 2004. pp. 186–195. IEEE, IEEE Computer Society Press, Rome, Italy (Oct, 17–19 2004)

[5] Barreto, P.S., Kim, H.Y., Lynn, B., Scott, M.: Efficient Algorithms for Pairing-Based Cryptosystems. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 354–369. Springer Berlin Heidelberg, Santa Barbara, USA (Aug 18–22, 2002)

[6] Bayer, S., Groth, J.: Efficient Zero-Knowledge Argument for Correctness of a Shuffle. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 263–280. Springer Berlin Heidelberg, Cambridge, UK (Apr 15–19, 2012)

[7] Bellare, M., Boldyreva, A., Palacio, A.: An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 171–188. Springer Berlin Heidelberg, Interlaken, Switzerland (May 2–6, 2004)

[8] Bellare, M., Garay, J.A., Rabin, T.: Batch Verification with Applications to Cryptography and Checking. In: Lucchesi, C.L., Moura, A.V. (eds.) LATIN'98: Theoretical Informatics. LNCS, vol. 1380, pp. 170–191. Springer Berlin Heidelberg, Campinas, Brazil (Apr 20–24, 1998)

[9] Bellare, M., Palacio, A.: GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active And Concurrent Attacks. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 162–177. Springer Berlin Heidelberg, Santa Barbara, USA (Aug 18–22, 2002)

[10] Benaloh, J., de Mare, M.: One-Way Accumulators: A Decentralized Alternative to Digital Signatures. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 274–285. Springer Berlin Heidelberg, Lofthus, Norway (May 23–27, 1993)

[11] Bitansky, N., Canetti, R., Paneth, O., Rosen, A.: On the Existence of Extractable One-Way Functions. In: Shmoys, D. (ed.) STOC 2014. pp. 505–514. ACM Press, New York, NY, USA (May 31 – Jun 3, 2014)

[12] Bitansky, N., Dachman-Soled, D., Garg, S., Jain, A., Kalai, Y.T., Lopez-Alt, A., Wichs, D.: Why "Fiat-Shamir for Proofs" Lacks a Proof. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 182–201. SSpringer Berlin Heidelberg, Tokyo, Japan (Mar 3–6, 2013)

[13] Blundo, C., Persiano, G., Sadeghi, A.R., Visconti, I.: Improved Security Notions and Protocols for Non-Transferable Identification. In: ESORICS 2008. pp. 364–378. Springer Berlin Heidelberg (2008)

[14] Boneh, D.: The Decision Diffie-Hellman Problem. In: P.Buhler, J. (ed.) The Algorithmic Number Theory Symposium. LNCS, vol. 1423, pp. 48–63. Springer Berlin Heidelberg, Portland, Oregon, USA (Jun 1998)

[15] Boneh, D., Boyen, X., Goh, E.J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer Berlin Heidelberg, Aarhus, Denmark (May 22–26, 2005)

[16] Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: Franklin, M.K. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer Berlin Heidelberg, Santa Barbara, USA (Aug 15–19, 2004)

[17] Bos, J.W., Costello, C., Naehrig, M.: Exponentiating in Pairing Groups. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 438–455. Springer Berlin Heidelberg, Burnaby, BC, Canada (Aug 14–16, 2013)

[18] Boudot, F.: Efficient Proofs That a Committed Number Lies in an Interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer Berlin Heidelberg, Bruges, Belgium (May 14–18, 2000)

[19] Brzuska, C., Farshim, P., Mittelbach, A.: Random-Oracle Uninstantiability from Indistinguishability Obfuscation. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015 (2). LNCS, vol. 9015, pp. 428–455. Springer Berlin Heidelberg, Warsaw, Poland (Mar 23–25, 2015)

[20] Camenisch, J., Chaabouni, R., shelat, a.: Efficient Protocols for Set Membership and Range Proofs. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 234–252. Springer Berlin Heidelberg, Melbourne, Australia (Dec 7–11, 2008)

[21] Camenisch, J., Lysyanskaya, A.: An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer Berlin Heidelberg, Innsbruck, Austria (May 6–10, 2001)

[22] Camenisch, J., Lysyanskaya, A.: Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer Berlin Heidelberg, Santa Barbara, USA (Aug 18–22, 2002)

[23] Canetti, R., Fischlin, M.: Universally Composable Commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer Berlin Heidelberg, Santa Barbara, USA (Aug 19–23, 2001)

[24] Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable Zero-Knowledge. In: STOC 2000. pp. 235–244. ACM Press, Portland, Oregon, USA (May 21–23 2000)

[25] Canetti, R., Goldreich, O., Halevi, S.: The Random Oracle Methodology, Revisited. In: Vitter, J.S. (ed.) STOC 1998. pp. 209–218. Dallas, Texas, USA (May 23–26, 1998)

[26] Chaabouni, R., Lipmaa, H., Zhang, B.: A Non-Interactive Range Proof with Constant Communication. In: Keromytis, A. (ed.) FC 2012. LNCS, vol. 7397, pp. 179–199. Springer Berlin Heidelberg, Bonaire, The Netherlands (Feb 27–Mar 2, 2012)

[27] Chaum, D.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communications of the ACM 24(2), 84–88 (1981)

[28] Chaum, D., Damgård, I.B., Van de Graaf, J.: Multiparty Computations Ensuring Privacy of Each Party's Input and Correctness of the Result. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 87–119. Springer Berlin Heidelberg, Santa Barbara, California, USA (16–20 Aug 1987)

[29] Ciampi, M., Persiano, G., Siniscalchi, L., Visconti, I.: A Transform for NIZK Almost as Efficient and General as the Fiat-Shamir Transform Without Programmable Random Oracles. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A (2). LNCS, vol. 9563, pp. 83–111. Springer Berlin Heidelberg, Tel Aviv, Israel (Jan 10–13, 2016)

[30] Danezis, G., Fournet, C., Groth, J., Kohlweiss, M.: Square Span Programs with Applications to Succinct NIZK Arguments. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014 (1). LNCS, vol. 8873, pp. 532–550. Springer Berlin Heidelberg, Kaohsiung, Taiwan, R.O.C. (Dec 7–11, 2014)

[31] De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 566–598. Springer Berlin Heidelberg, Santa Barbara, USA (Aug 19–23, 2001)

[32] De Santis, A., Micali, S., Persiano, G.: Non-Interactive Zero-Knowledge Proof Systems. In: CRYPTO 1987. pp. 52–72. Springer Berlin Heidelberg (1987)

[33] Di Crescenzo, G., Katz, J., Ostrovsky, R., Smith, A.: Efficient and Non-Interactive Non-Malleable Commitment. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 40–59. Springer Berlin Heidelberg, Innsbruck, Austria (May 6–10, 2001)

[34] Diffie, W., Hellman, M.E.: New Directions in Cryptography. IEEE transactions on Information Theory 22(6), 644–654 (1976)

[35] El Gamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In: Blakley, G.R., Chaum, D. (eds.) Advances in Cryptology: Proceedings of CRYPTO 84. LNCS, vol. 196, pp. 10–18. Springer Berlin Heidelberg, Santa Barbara, California, USA (Aug 19–22, 1984)

[36] Elkin, M.: An Improved Construction of Progression-free Sets. In: Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms. pp. 886–905. Society for Industrial and Applied Mathematics (2010)

[37] Escala, A., Groth, J.: Fine-Tuning Groth-Sahai Proofs. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 630–649. Springer Berlin Heidelberg, Buenos Aires, Argentina (March 26–28, 2014)

[38] Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.L.: An Algebraic Framework for Diffie-Hellman Assumptions. In: Canetti, R., Garay, J. (eds.) CRYPTO (2) 2013. LNCS, vol. 8043, pp. 129–147. Springer Berlin Heidelberg, Santa Barbara, California, USA (Aug 18–22, 2013)

[39] Fauzi, P., Lipmaa, H.: Efficient Culpably Sound NIZK Shuffle Argument without Random Oracles. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 200–216. Springer International Publishing, San Franscisco, CA, USA (February 29–March 4, 2016)

[40] Fauzi, P., Lipmaa, H., Zając, M.: A Shuffle Argument Secure in the Generic Model. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016 (2). LNCS, vol. 10032, pp. 841–872. Springer Berlin Heidelberg, Hanoi, Vietnam (Dec 4–8 2016)

[41] Fauzi, P., Lipmaa, H., Zhang, B.: Efficient Modular NIZK Arguments from Shift and Product. In: Abdalla, M., Nita-Rotaru, C., Dahab, R. (eds.) CANS 2013. LNCS, vol. 8257, pp. 92–121. Springer International Publishing, Paraty, Brazil (Nov 20–22, 2013)

[42] Fauzi, P., Lipmaa, H., Zhang, B.: Efficient Non-Interactive Zero Knowledge Arguments for Set Operations. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 216–233. Springer Berlin Heidelberg, Bridgetown, Barbados (March 3–7, 2014)

[43] Feige, U., Fiat, A., Shamir, A.: Zero-knowledge Proofs of Identity. Journal of Cryptology 1(2), 77–94 (1988)

[44] Feige, U., Shamir, A.: Witness Indistinguishable and Witness Hiding Protocols. In: Proceedings of the twenty-second annual ACM symposium on Theory of computing. pp. 416–426. ACM (1990)

[45] Fiat, A., Shamir, A.: How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In: CRYPTO 1986. pp. 186–194. Springer Berlin Heidelberg (1986)

[46] Fischlin, M.: Trapdoor Commitment Schemes and Their Applications. Ph.D. thesis, Citeseer (2001)

[47] Furukawa, J.: Efficient and Verifiable Shuffling and Shuffle-Decryption. IEICE Transactions 88-A(1), 172–188 (2005)

[48] Gennaro, R., Gentry, C., Parno, B.: Non-Interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer Berlin Heidelberg, Santa Barbara, California, USA (Aug 15–19, 2010)

[49] Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic Span Programs and NIZKs without PCPs. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645. Springer Berlin Heidelberg, Athens, Greece (Apr 26–30, 2013)

[50] Goldreich, O., Goldwasser, S., Micali, S.: On the cryptographic applications of random functions (extended abstract). In: Blakley, G.R., Chaum, D. (eds.) Advances in Cryptology: Proceedings of CRYPTO 84. LNCS, vol. 196, pp. 276–288. Springer Berlin Heidelberg, Santa Barbara, California, USA (Aug 19–22, 1984)

[51] Goldreich, O., Mansour, Y., Sipser, M.: Interactive proof systems: Provers that never fail and random selection. In: Foundations of Computer Science, 1987., 28th Annual Symposium on. pp. 449–461. IEEE Computer Society (1987)

[52] Goldwasser, S., Kalai, Y.T.: On the (In)security of the Fiat-Shamir Paradigm. In: FOCS 2003. pp. 102–113. IEEE, IEEE Computer Society Press, Cambridge, MA, USA (Oct 11–14, 2003)

[53] Goldwasser, S., Micali, S.: Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. In: Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing. pp. 365–377. ACM, San Francisco, California, USA (May 5–7, 1982)

[54] Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof Systems. SIAM J. Comput. 18(1), 186–208 (1989)

[55] Groth, J.: A Verifiable Secret Shuffle of Homomorphic Encryptions. J. Cryptology 23(4), 546–579 (2010)

[56] Groth, J.: Short Pairing-Based Non-interactive Zero-Knowledge Arguments. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340. Springer Berlin Heidelberg, Singapore (Dec 5–9, 2010)

[57] Groth, J.: On the Size of Pairing-based Non-interactive Arguments. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 305–326. Springer Berlin Heidelberg, Vienna, Austria (May 8–12, 2016)

[58] Groth, J., Lu, S.: A Non-interactive Shuffle with Pairing Based Verifiability. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 51–67. Springer Berlin Heidelberg, Kuching, Malaysia (Dec 2–6, 2007)

[59] Groth, J., Ostrovsky, R., Sahai, A.: New Techniques for Noninteractive Zero-Knowledge. Journal of the ACM 59(3) (2012)

[60] Groth, J., Sahai, A.: Efficient Noninteractive Proof Systems for Bilinear Groups. SIAM J. Comput. 41(5), 1193–1232

[61] Henry, R., Goldberg, I.: All-but-k Mercurial Commitments and their Applications. Tech. Rep. 26, Centre for Applied Cryptographic Research (Dec 2012), available at `http://cacr.uwaterloo.ca/techreports/2012/cacr2012-26.pdf`

[62] Jarecki, S., Liu, X.: Efficient Oblivious Pseudorandom Function with Applications to Adaptive OT and Secure Computation of Set Intersection. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 577–594. Springer Berlin Heidelberg, San Francisco, CA, USA (Mar 15–17 2009)

[63] Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-Size Commitments to Polynomials and Their Applications. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 177–194. Springer Berlin Heidelberg, Singapore (Dec 5–9, 2010)

[64] Kilian, J.: A Note on Efficient Zero-Knowledge Proofs and Arguments. In: Proceedings of the Twenty-Fourth Annual ACM Symposium on the Theory of Computing. pp. 723–732. Victoria, British Columbia, Canada (4–6 May 1992)

[65] Kissner, L., Song, D.: Privacy-Preserving Set Operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer Berlin Heidelberg, Santa Barbara, USA (Aug 14–18, 2005)

[66] Lindell, Y.: An Efficient Transform from Sigma Protocols to NIZK with a CRS and Non-programmable Random Oracle. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015 (1). LNCS, vol. 9014, pp. 93–109. Springer Berlin Heidelberg, Warsaw, Poland (Mar 23–25, 2015)

[67] Lipmaa, H.: Progression-Free Sets and Sublinear Pairing-Based Non-Interactive Zero-Knowledge Arguments. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 169–189. Springer Berlin Heidelberg, Taormina, Italy (Mar 18–21, 2012)

[68] Lipmaa, H.: Efficient NIZK Arguments via Parallel Verification of Benes Networks. In: Abdalla, M., de Prisco, R. (eds.) SCN 2014. LNCS, vol. 8642, pp. 416–434. Springer International Publishing, Amalfi, Italy (September 3–5, 2014)

[69] Lipmaa, H.: Prover-Efficient Commit-And-Prove Zero-Knowledge SNARKs. In: Pointcheval, D., Nitaj, A., Rachidi, T. (eds.) AFRICACRYPT

2016. LNCS, vol. 9646, pp. 185–206. Springer International Publishing, Fes, Morocco (Apr 13–15, 2016)

[70] Lipmaa, H., Asokan, N., Niemi, V.: Secure Vickrey Auctions without Threshold Trust. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 87–101. Springer berlin Heidelberg, Southhampton Beach, Bermuda (Mar 11–14, 2002)

[71] Lipmaa, H., Zhang, B.: A More Efficient Computationally Sound Non-Interactive Zero-Knowledge Shuffle Argument. Journal of Computer Security 21(5), 685–719 (2013)

[72] Maurer, U.M.: Abstract Models of Computation in Cryptography. In: Smart, N.P. (ed.) WCC 2005. LNCS, vol. 3796, pp. 1–12. Springer Berlin Heidelberg, Cirencester, UK (Dec 19–21, 2005)

[73] McCurley, K.S.: The Discrete Logarithm Problem. In: Proc. of Symp. in Applied Math. vol. 42, pp. 49–74. USA (1990)

[74] Morillo, P., Ràfols, C., Villar, J.L.: The Kernel Matrix Diffie-Hellman Assumption. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016 (1). LNCS, vol. 10031, pp. 729–758. Springer Berlin Heidelberg, Hanoi, Vietnam (Dec 4–8 2016)

[75] Naor, M.: On Cryptographic Assumptions and Challenges. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 96–109. Springer Berlin Heidelberg, Santa Barbara, USA (Aug 17–21, 2003)

[76] Parno, B., Gentry, C., Howell, J., Raykova, M.: Pinocchio: Nearly Practical Verifiable Computation. In: IEEE SP 2013. pp. 238–252. IEEE Computer Society, Berkeley, CA, USA (May 19-22, 2013)

[77] Pavan, A., Santhanam, R., Vinodchandran, N.: Some Results on Average-Case Hardness within the Polynomial Hierarchy. In: FSTTCS 2006. pp. 188–199. Springer Berlin Heidelberg (2006)

[78] Pedersen, T.P.: Non-Interactive And Information-Theoretic Secure Verifiable Secret Sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer Berlin Heidelberg, Santa Barbara, California, USA (Aug 11–15, 1991)

[79] Pfitzmann, A., Waidner, M.: Networks Without User Observability - Design Options. In: EUROCRYPT 1985. pp. 245–253. Springer Berlin Heidelberg (1985)

[80] Rivest, R.L., Adleman, L., Dertouzos, M.L.: On Data Banks and Privacy Homomorphisms. Foundations of Secure Computation 4(11), 169–180 (1978)

[81] Schwartz, J.T.: Fast Probabilistic Algorithms for Verification of Polynomial Identities. Journal of the ACM 27(4), 701–717 (1980)

[82] Syropoulos, A.: Mathematics of Multisets. In: Multiset Processing. pp. 347–358. Springer Berlin Heidelberg (2001)

[83] Tao, T., Vu, V.: Additive Combinatorics. Cambridge Studies in Advanced Mathematics, Cambridge University Press (2006)

[84] Tsiounis, Y., Yung, M.: On the Security of ElGamal Based Encryption. In: International Workshop on Public Key Cryptography. pp. 117–134. Springer Berlin Heidelberg (1998)

[85] Yuen, T.H., Huang, Q., Mu, Y., Susilo, W., Wong, D.S., Yang, G.: Efficient Non-interactive Range Proof. In: Ngo, H.Q. (ed.) COCOON 2009. LNCS, vol. 5609, pp. 138–147. Springer, Heidelberg, Niagara Falls, NY, USA (Jul 13–15, 2009)

[86] Zippel, R.: Probabilistic Algorithms for Sparse Polynomials. In: Ng, E.W. (ed.) EUROSM 1979. LNCS, vol. 72, pp. 216–226. Springer Berlin Heidelberg, Marseille, France (Jun 1979)

ACKNOWLEDGMENTS

First of all I would like to thank my supervisor Helger Lipmaa for all the aid, mentoring, and research suggestions he has generously given throughout my PhD studies. You showed me what it takes to succeed in academia and how to get there, and for that I am eternally grateful. I would also like to thank my other co-authors Bingsheng Zhang and Michał Zając, and other professors and colleagues in the university for all their engaging discussions that have led not only to interesting research papers, but also, more importantly, to an improvement in my overall knowledge.

I am grateful to have excellent reviewers whose feedback has greatly improved the quality of this thesis. For this I thank the University of Tartu internal reviewer Sven Laur, and pre-reviewers Ivan Visconti and Aggelos Kiayias.

Throughout my studies, I was funded by the the Estonian Research Council under research grant IUT2-1, and the EU's Horizon 2020 research and innovation programme under grant agreement No 653497 (project PANORAMIX). I was also supported by the Estonian ICT Doctoral School, and the Estonian IT academy. My sincere gratitude to all concerned.

I would not be here without the love and support of my family. My parents who have blessed me with both a name and environment that encouraged me to read and always want to learn more. My siblings, cousins, and other family members who have given me encouragement and useful advice. You have all inspired me to be a better person. *Terima kasih banyak, keluargaku.*

Last but not least, I would like to thank all my wonderful friends, spread all across the globe, who have helped me along the way, and made me smile when I needed it the most. I won't give names to avoid adding several more pages, but you know who you are. And a special mention to the Tartu Ultimate Frisbee club, for giving me an outlet to stay healthy and in high spirits, while also gaining me some friends for life.

KOKKUVÕTE (SUMMARY IN ESTONIAN)

EFEKTIIVSED MITTEINTERAKTIIVSED NULLTEADMUSPROTOKOLLID REFERENTSSÕNE MUDELIS

Digitaalse ajastu võidukäiguga on interneti vahendusel võimalik sooritada aina ulmelisemaid tegevusi. Täielikule krüpteeringule ehitatud mobiilsed rakendused, nagu näiteks WhatsApp, suudavad tagada kõne või sõnumi jõudmise üksnes õige adressaadini. Enamik pangasüsteeme garanteerivad TLS protokolli kasutades, et arvete maksmisel ja ülekannete sooritamisel ei oleks tehingute andmeid kellelgi võimalik lugeda ega muuta. Mõned riigid pakuvad võimalust elektroonilisel teel hääletada (näiteks Eesti) või referendumeid läbi viia (näiteks Šveits), tagades sealjuures traditsioonilise paberhääletuse tasemel turvalisuse kriteeriumid. Kõik eelnevalt kirjeldatud tegevused vajavad kasutajate turvalisuse tagamiseks krüptograafilisi protokolle.

Tegelikkuses ei ole võimalik eeldada, et kõik protokolli osapooled järgivad protokolli kirjeldust. Reaalses elus peab protokolli turvalisuseks iga osapool tõestama, et ta protokolli kirjeldust järgis. Paraku on osapoolel tihti ilma oma sisendi privaatsust ohverdamata seda väga raske teha. Üks viis selleks on kasutada nullteadmusprotokolli. Nullteadmusprotokolliks nimetatakse tõestust, mis ei lekita muud teavet peale selle, et väide on tõene. Lisades piirangu, et tõestaja töötab polüomiaalse ajaga, saame nullteadusargumendi.

Tihti soovime, et nullteadmusargument oleks mitteinteraktiivne ja edasikantav. Sellisel juhul piisab, kui tõestus on arvutatud ainult ühe korra ning verifitseerijatel on võimalik seda igal ajal kontrollida. Interaktiivset nullteadmusprotokolli on võimalik muuta mitteinteraktiivseks kasutades (näiteks) Fiat-Shamiri heuristikat, kus verifitseerija sõnumid tõestajale on asendatud juhuslike sõnumitega samast hulgast. Saadud protokoll on turvaline juhusliku oraakli mudelis.

Teine viis mitteinteraktiivsuse saavutamiseks on kasutada referentssõne mudelit (inglise keeles *reference string model*), kus usaldusväärne kolmas osapool väljastab ühise sõne, mis sisaldab ausa verifitseerija vastust interaktiivses protokollis. Käesolevas töös pakume välja kolm stsenaariumit, kus mitteinteraktiivne nullteadmusargument referentssõne mudelis on efektiivsuselt võrreldav parimate teadaolevate mitteinteraktiivsete argumentidega juhusliku oraakli mudelis.

Esimeses stsenaariumis vaatleme verifitseeritavat arvutamist ning seda, kuidas lihtsalt kontrollida $\mathbf{NP}$-täielike ülesannete lahendusi. Selleks koostame kaks põhilist mitteinteraktiivset argumenti referentssõne mudelis ja näitame, et nende abil on võimalik koostada mitteinteraktiivne argument mistahes $\mathbf{NP}$ keele jaoks, sealhulgas efektiivne intervallargument. Saadud mitteinteraktiivsed nullteadmusargumendid $\mathbf{NP}$ keelte jaoks ja intervallargumendid on efektiivsemad kui varasemad tulemused referentssõne mudelis ning kergemini kontrollitavad kui CIRCUIT-SAT.

Teises stsenaariumis uurime autoriseerimise väljakutset reaalses elus. Tasub tähele panna, et mitmed sellised situatsioonid sisaldavad tõestust mingile hulkadevahelisele seosele ja privaatsus vajab ühe või mitme hulga peitmist relatsioonis. Konstrueerime uue mitteinteraktiivse nullteadmusargumendi, et tõestada lihtne relatsioon PMSET nelja kinnitatud multihulga vahel ja näitame, kuidas neid kasutada hulga sisalduvuse ja intervall- ning teiste argumentide konstrueerimiseks. Sellest tuletatud mitteinteraktiivsed nullteadmusargumendid hulga tehete jaoks on sama efektiivsed kui varasemad tulemused juhusliku oraakli mudelis, kuid suurema tehete hulgaga.

Kolmandas stsenaariumis vaatleme krüptogrammide segamist enne dekrüpteerimist kui viisi privaatsuse tõstmiseks elektroonilise hääletuse süsteemides, peites ära seose hääletaja ja tema hääle vahel. Konstrueerime kaks efektiivset segamisargumenti referentssõne mudelis, mis on efektiivsuselt peaaegu võrdsed varasemate tulemustega juhusliku oraakli mudelis.

CURRICULUM VITAE

Personal data

Name	Prastudy Mungkas Fauzi
Birth	January 31st, 1986
Citizenship	Indonesian
Marital Status	Single
Languages	Indonesian, English
Contact	prastudy.fauzi@gmail.com

Education

2012–	University of Tartu, Ph.D. student in Computer Science
2010–2012	University of Tartu, M.Sc. in Engineering (Comp. Sci.), and Norwegian University of Science and Technology, M.Sc. in Security and Mobile Computing
2003–2007	University of Indonesia, B.Sc. (Comp. Sci.)
2000–2003	SMA Negeri 1 Mataram, Indonesia, secondary education
1998–2000	SMP Negeri 2 Mataram, Indonesia, secondary education
1994–1998	Ironside State School, Australia, primary education

Employment

2012–2014	University of Tartu, assistant in informatics (0.5)
2011–2011	Altrovis, web developer
2010–2010	PT Mitra Tri-atma, SharePoint engineer
2009–2010	Javanex Media System, web developer
2008–2009	Pusat Ilmu Komputer UI, web developer

ELULOOKIRJELDUS

Isikuandmed

Nimi	Prastudy Mungkas Fauzi
Sünniaeg ja -koht	31. jaanuar 1986
Kodakondsus	indoneeslane
Perekonnaseis	vallaline
Keelteoskus	indoneesia, inglise
Kontaktandmed	prastudy.fauzi@gmail.com

Haridustee

2012–	Tartu Ülikool, informaatika doktorant
2010–2012	Tartu Ülikool, tehnikateaduse magister (informaatika) Norra Teadus ja Tehnika Ülikool, magister turvalisuses ja mobiilses arvutamises,
2003–2007	Indonesia Ülikool, bakalaureus infotehnoloogias
2000–2003	SMA Negeri 1 Mataram, Indoneesia, keskharidus
1998–2000	SMP Negeri 2 Mataram, Indoneesia, keskharidus
1994–1998	Ironside'i riiklik kool, Austraalia, põhiharidus

Teenistuskäik

2012–2014	Tartu Ülikool, informaatika assistent (0.5)
2011–2011	Altrovis, veebiarendaja
2010–2010	PT Mitra Tri-atma, SharePointi arendaja
2009–2010	Javanex Media System, veebiarendaja
2008–2009	Pusat Ilmu Komputer UI, veebiarendaja

20. **Jüri Lember.** Consistency of empirical k-centres. Tartu, 1999, 148 p.
21. **Ella Puman.** Optimization of plastic conical shells. Tartu, 2000, 102 p.
22. **Kaili Müürisep.** Eesti keele arvutigrammatika: süntaks. Tartu, 2000, 107 lk.
23. **Varmo Vene.** Categorical programming with inductive and coinductive types. Tartu, 2000, 116 p.
24. **Olga Sokratova.** Ω-rings, their flat and projective acts with some applications. Tartu, 2000, 120 p.
25. **Maria Zeltser.** Investigation of double sequence spaces by soft and hard analitical methods. Tartu, 2001, 154 p.
26. **Ernst Tungel.** Optimization of plastic spherical shells. Tartu, 2001, 90 p.
27. **Tiina Puolakainen.** Eesti keele arvutigrammatika: morfoloogiline ühestamine. Tartu, 2001, 138 p.
28. **Rainis Haller.** *M(r,s)*-inequalities. Tartu, 2002, 78 p.
29. **Jan Villemson.** Size-efficient interval time stamps. Tartu, 2002, 82 p.
30. Töö kaitsmata.
31. **Mart Abel.** Structure of Gelfand-Mazur algebras. Tartu, 2003. 94 p.
32. **Vladimir Kuchmei.** Affine completeness of some ockham algebras. Tartu, 2003. 100 p.
33. **Olga Dunajeva.** Asymptotic matrix methods in statistical inference problems. Tartu 2003. 78 p.
34. **Mare Tarang.** Stability of the spline collocation method for volterra integro-differential equations. Tartu 2004. 90 p.
35. **Tatjana Nahtman.** Permutation invariance and reparameterizations in linear models. Tartu 2004. 91 p.
36. **Märt Möls.** Linear mixed models with equivalent predictors. Tartu 2004. 70 p.
37. **Kristiina Hakk.** Approximation methods for weakly singular integral equations with discontinuous coefficients. Tartu 2004, 137 p.
38. **Meelis Käärik.** Fitting sets to probability distributions. Tartu 2005, 90 p.
39. **Inga Parts.** Piecewise polynomial collocation methods for solving weakly singular integro-differential equations. Tartu 2005, 140 p.
40. **Natalia Saealle.** Convergence and summability with speed of functional series. Tartu 2005, 91 p.
41. **Tanel Kaart.** The reliability of linear mixed models in genetic studies. Tartu 2006, 124 p.
42. **Kadre Torn.** Shear and bending response of inelastic structures to dynamic load. Tartu 2006, 142 p.
43. **Kristel Mikkor.** Uniform factorisation for compact subsets of Banach spaces of operators. Tartu 2006, 72 p.