UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Advanced cryptographic techniques for building verifiable and transparent electronic voting protocols

Alex Escala Ribas

Universitat Politècnica de Catalunya

Doctoral Programme in Applied Mathematics

Department of Mathematics

Advanced Cryptographic Techniques for Building Verifiable and Transparent Electronic Voting Protocols

PhD Dissertation

Alex Escala Ribas

Supervisor: Dr. Maria Paz Morillo Bosch

May 2017

Abstract

Electronic voting presents many challenges due to its multiple security requirements. Some of the challenges are related to guaranteeing voters' privacy and system's transparency, which are hard to satisfy simultaneously. Electronic voting also presents other challenges such as usability, particularly from the voter's side.

We study two problems of electronic voting. Cast-as-intended verifiability comprises those mechanisms which assure the voter that her cast ballot corresponds to her chosen voting option. Current proposals put the verification burden on the voter, something which is undesirable in real-world elections, where both technically skilled and non-skilled voters participate. In this thesis, we introduce the concept of universal cast-as-intended verifiability, which provides mechanisms which allow any entity to check that any ballot corresponds to the voter's selections - without revealing them. We formally define what universal cast-as-intended verifiability is and we give an electronic voting protocol satisfying this property.

The other problem we have studied is the problem of invalid votes in electronic elections. Since a common selling point of electronic voting is that it avoids voters inadvertently spoiling their votes, deliberately spoiled ballots appearing in the tallying phase of an electronic election can cause mistrust on the system. Indeed, election stakeholders might think that the system is flawed or that it was exploited somehow. To avoid this situation, we define the concept of vote validatability, which states the electronic voting system should be able to detect spoiled ballots before they are successfully cast. In addition to formally defining this notion, we design an electronic voting protocol satisfying this property.

All these security requirements of electronic voting systems are implemented with cryptographic tools. In addition to encryption and signature schemes, another essential primitive for building electronic voting protocols is zero-knowledge proofs. Zero-knowledge proofs allow a prover to convince a verifier that a statement is true without leaking any other information. These zero-knowledge proofs can be used to, for example, prove that the tally of the election was done properly. Recently, Groth and Sahai constructed efficient non-interactive zero-knowledge proofs for a wide range of statements including, among others, statements appearing in electronic voting.

In this thesis, we give two contributions on Groth-Sahai proofs. On the one hand, we give a framework for deriving cryptographic assumptions from which to build secure cryptographic protocols. In particular, we build new Groth-Sahai proofs improving the efficiency of currently known constructions. Independently, we show how the original Groth-Sahai proofs can be extended to be compatible with even more statements, how to improve their out-of-the-box efficiency for many of these statements and how to improve their re-usability efficiency among multiple statements.

***Keywords*—** Electronic Voting, Verifiability, Cryptography, Zero-Knowledge Proofs

Resum

Els sistemes de vot electrònic presenten molts reptes a causa dels seus múltiples requeriments. Alguns reptes són garantir la privacitat del votant i la transparència del sistema, difícils de satisfer al mateix temps. Els sistemes de vot electrònic també presenten reptes com la usabilitat, sobretot de cara als votants.

En aquesta tesi estudiem dos problemes del vot electrònic. La verificabilitat *cast-as-intended* tracta d'obtenir mecanismes que garanteixin al votant que el seu vot correspon a les seves preferències. Les propostes actuals posen la càrrega de la verificació en el votant, cosa que no és desitjable en eleccions del món real, en les que participen votants amb diferents graus de coneixements tècnics. Nosaltres introduïm el concepte de *universal cast-as-intended verifiability*, que proporciona mecanismes perquè qualsevol entitat de l'elecció pugui comprovar que qualsevol vot conté les preferències del votant que l'ha emès - sense revelar el contingut del vot. A banda de definir formalment el concepte de *universal cast-as-intended verifiability*, també proposem un protocol de vot electrònic que satisfà aquesta propietat.

L'altre problema que hem estudiat és el problema dels vots invàlids en eleccions electròniques. Un dels avantatges del vot electrònic és que permet evitar que els votants emetin vots nuls sense voler. Per això, si durant el recompte de l'elecció apareixen vots nuls construïts intencionadament es pot crear desconfiança en el sistema de vot. Els usuaris del sistema de vot poden pensar que el sistema té forats de seguretat o que ha estat atacat. Per evitar aquesta situació, definim el concepte de *vote validatability*, una propietat dels sistemes de vot electrònic que garanteix que els vots nuls es poden identificar en el moment en què s'emeten. En aquesta tesi hem definit formalment aquesta propietat i hem dissenyat un protocol que la satisfà.

Tots aquests requisits de seguretat dels protocols de vot electrònic s'implementen amb eines criptogràfiques. Les principals eines que s'utilitzen són esquemes de xifrat, esquemes de firma i proves de coneixement zero. Una prova de coneixement zero permet a una entitat convèncer una altra entitat que una sentència és certa sense donar cap altra informació que la certesa de la sentència. Aquestes proves de coneixement zero es poden fer servir, per exemple, per demostrar que el recompte de l'elecció s'ha fet correctament. Recentment, Groth i Sahai han construït proves de coneixement zero que es poden fer servir per un ampli ventall de sentències com les que apareixen en protocols de vot electrònic.

En aquesta tesi hem fet dos contribucions sobre les proves de Groth i Sahai. Per una banda, donem un marc teòric que permet derivar hipòtesis criptogràfiques per construir protocols criptogràfics. En particular, construïm noves proves de Groth i Sahai millorant l'eficiència de les construccions existents. De manera independent, indiquem com les proves de Groth i Sahai es poden estendre per fer-les compatibles amb un ventall més ampli de sentències, millorem l'eficiència de les proves de Groth i Sahai per moltes sentències i millorem l'eficiència de les proves de Groth i Sahai quan es fan servir per demostrar múltiples sentències.

Paraules clau— Vot Electrònic, Verificabilitat, Criptografia, Proves de Coneixement Zero

Agraïments

Començaré els agraïments de l'única manera que em sembla justa. Moltes gràcies, Paz! Aquesta tesi representa la culminació d'una feina que vam començar fa ben bé nou anys, quan jo entrava com a becari al departament. L'únic que puc fer després de tant de temps és agrair-te tot el que m'has ensenyat i tota la paciència que has tingut. Sense tu aquesta tesi no hauria estat possible.

Voldria seguir per donar les gràcies al que, podríem dir, va ser el meu mecenes pel que fa a la recerca. Gràcies, Miquel! En uns temps en què viure de la recerca no és gens fàcil, tu em vas permetre, de manera totalment desinteressada, dedicar-m'hi a temps complet.

Thank you, Jens, for accepting me as your student at UCL and for being my master's thesis advisor. I learnt a lot both from your exceptional lectures and from your invaluable advice.

Durant el temps que vaig estar pel departament MA4 no només vaig aprendre coses de la meva tutora. Vaig tenir la sort de tenir uns companys extraordinaris, amb qui vaig passar bones estones i amb qui vaig treballar molt (i molt a gust). Gràcies, Carla, Javi i Jorge!

També voldria donar les gràcies als companys i amics que vaig fer a Scytl. Gràcies al Jordi per donar-me l'oportunitat de treballar-hi i gràcies als meus companys de departament, especialment a la Sandra, amb qui vaig aprendre com funciona això del vot electrònic.

Aprofito per donar gràcies a tota la gent amb qui he treballat i que encara no he anomenat: gràcies, Benoît, Gottfried, Eike, Jordi, Guillermo, Víctor i Pedro!

He tingut la sort de trobar dos revisors externs que han tingut el valor de revisar aquesta tesi. Gràcies, Joan i Josep Maria!

L'última etapa d'aquesta tesi no hauria estat el mateix sense els meus companys de Kernel, que m'han donat el suport que necessitava per poder acabar d'escriure-la. Gràcies!

I finalment, però no per això menys important, voldria donar les gràcies a la meva família, i en especial a l'Eli, per haver estat sempre al meu costat durant tot aquest temps.

Contents

Chapter 1

Introduction

1.1 Why is electronic voting so difficult?

The society we live in is constantly changing due to fast progress in technology. Technology has changed how we interact with our bank, how we shop or how we interact with our government. Perhaps the most challenging aspect of society that is changing is how we vote.

In traditional elections, voting is done during (usually) one day. During this day, voters go to a polling place, put a piece of paper containing their selections into an envelope and deposit the envelop inside a box. After the voting period, the envelopes in the box are shuffled, each envelope is opened and all the selections are counted. People are generally satisfied with this process because they have high confidence that the results announced by the election authorities reflect the voters' will.

This confidence is built through different mechanisms. First, the voter can inspect the contents of the paper which she is placing inside the envelope. The voter can also check that her envelope is deposited inside the box. These boxes are transparent so that voters can always see their content. In addition, during the entire voting day there are several people verifying that the whole process is being done correctly. These auditors, who are affiliated to different political parties and non-governmental organizations, are in charge of checking that the ballot box is empty at the beginning of the election, that only voters who are in the electoral roll deposit envelopes inside the ballot box, that no ballots are removed and that the counting process is done correctly. If trust is put on the auditors, all these mechanisms give sufficient guarantees that the result of the election exactly corresponds to the voters' will.

When technology is included in the electronic voting process, some of these mechanisms can no longer be used. Let's take as example the most extreme scenario in which voters cast their votes from their personal devices (computer, smart phone, tablet or other). In this case, voters select their voting option and let the voting device construct their digital ballot. These digital ballots are sent via the Internet and are stored inside a digital ballot box. At the end of the election, all the votes inside the digital ballot box are counted by a computer, which outputs a result.

In this digital scenario, all the mechanisms which gave confidence to the voter that the election process was done correctly no longer make sense. For example, the voter has no means of checking whether there is a digital ballot corresponding to her cast vote stored in the digital ballot box. Similarly, the process of counting the votes is completely opaque to any entity.

This lack of transparency is common to many processes which have been transformed by technology. For example, when buying through the Internet the buyer has no means to tell whether the amount of money sent to the seller equals what she saw advertised in the digital store. However, the bank can raise an alarm if the buyer transfers some unusually high amount of money. This measure is easy to implement since the bank *knows* how much money the buyer is spending in each transaction. In addition, banks have insurances which might cover (part of) the fraud which can occur in electronic payments.

There is one key issue which makes electronic voting different from the other scenarios. Only the voter should know what her chosen voting option is. This privacy requirement creates a lot of challenges by itself. Designing electronic voting systems which guarantee the privacy of the vote is an active research topic. Things get even more challenging when the goal is not only to provide privacy but also to bring back those mechanisms which give confidence to the voter that the election result is the correct one.

It is not enough to have an electronic voting system which provides the same privacy and transparency guarantees as traditional voting systems. There are other challenges which need to be solved to have electronic voting systems which people trust. One of such challenges is related to how invalid or null votes are processed. In traditional voting systems, voters can easily spoil ballots by, for example, making selections which are not allowed. One of such cases would be to make two selections in a single-mark ballot. During the tallying of the ballots, such ballots are identified as invalid and they are excluded from the result. This is a well-understood process and it is usual to see elections in which a small percentage of ballots are invalid.

In electronic voting systems, voters can also spoil ballots. Even though the visual interface which is used to digitally cast a ballot can prevent casting spoiled votes, a technically skilled voter can overcome such protections by creating a ballot using a maliciously crafted voting algorithm. Following the analogy with traditional voting, such spoiled vote would be detected in the tallying phase of the election. One would expect that detecting and rejecting the spoiled vote would be a satisfactory approach.

However, detecting spoiled votes at the tallying phase creates confusion and mistrust among electoral administrators. One of the reasons is that there is the common belief that one of the advantages of electronic voting is that it can prevent having spoiled votes. This creates the belief that a spoiled vote *must* be caused by either a bug or an exploit of the system. Similarly, a voter claiming that she was able to cast a spoiled vote might be perceived as a sign of a flaw in the electronic voting system.

Therefore, a requirement that one might expect from electronic voting systems is that no invalid votes can be cast. In other words, the voting system should not accept a ballot unless it corresponds to a valid vote. And, of course, this should be done without compromising the security of the ballots.

The issue of invalid votes is particularly interesting. On the one hand, from an academic point of view it is not regarded as a problem (after all, an invalid vote will not count towards the result of the election). In particular, no research has been done to solve this issue. On the other hand, this is a real issue in the *real world*, where it is not enough to have a secure voting system but the voting system also needs to be trusted by the stakeholders of an election.

Last, but not least, the electronic voting system needs to be *efficient*. This means that, for example, tallying the election should take a reasonable amount of time. Reducing the time it takes to tally the election is indeed one of the selling points of electronic voting. But the efficiency requirement also means that casting a vote should take a reasonable time as voters expect not to wait a long time once they have clicked the "cast vote" button. Satisfying the privacy and transparency requirements while keeping the computational cost of casting a vote low is another of the challenges that appear when building electronic voting systems.

Efficiency requirements are sometimes overlooked in academic proposals, where the exact time it takes to cast a ballot is usually not relevant. On the other hand, electronic voting systems used in real elections should be designed so that casting a vote takes at most a few seconds.

1.2 Privacy, verifiability and validatability

Classically, security requirements of electronic voting systems have been divided in two classes: those which are related to privacy and those which are related to verifiability. In addition, in this thesis we introduce the requirement of vote validatability, which states that it should not be possible to get a ballot corresponding to an invalid vote inside the digital ballot box. In this section, we give some more details on these requirements.

1.2.1 Privacy requirements

A basic requirement of any electronic voting system is that no stakeholder of the election, including the administrators of the voting system, can know what voting option was chosen by any individual voter. In most of the electronic voting systems, the voter casts her vote by introducing the chosen voting option in a voting device. This implies that we need to trust the voting device to not leak the voting option chosen by the voter, otherwise we can get no privacy guarantee. However, this is one of the few assumptions that is made regarding privacy.

There are proposals of electronic voting schemes in which the voter does not introduce her voting option into the voting device. In these systems, the voter introduces some codes related to the chosen voting option. The relation between codes and voting options is unknown to the voting device, so it is no longer needed to assume that the voting device will not leak the voters' selection. Despite this property of such voting systems, there are usability problems in having to introduce many codes instead of selecting a candidate through a graphical user interface, which is why other types of voting schemes are more popular.

Going back to schemes in which the voting device is trusted, we can refine the notion of privacy. It is commonly assumed that a private electronic voting scheme should not give any receipt to the voter which can be used to sell her vote. In other words, the process of casting a vote should not output any information which can be easily traced back to the voting option chosen by the voter. However, when academics talk about receipt-freeness they refer to another security requirement.

Receipt-freeness is a stronger notion of privacy. It essentially states that no voter should be able to produce a ballot in a way that she can later prove to someone else what voting option corresponds to the ballot. This means that a voter should not be able to show how she voted when she uses the intended method for casting a ballot but that this should hold even if the voter deviates from the protocol and uses any other method for casting a ballot.

Finally, there are other stronger notions of privacy. A coercion-resistant scheme should not only hide the voting option chosen by the voter but it should also hide the fact that the voter participated in the election. This should hold even if someone was actively trying to prevent the voter from participating in the election or trying to force the voter to vote in a particular way. There is always a trade-off between usability and privacy. In particular, existing coercion-resistant electronic voting schemes are completely unusable.

Satisfying strong notions of privacy is a hard challenge by itself. The challenge gets even harder when one wants to consider verifiability properties.

1.2.2 Verifiability requirements

To build trust on the electronic voting system, transparency requirements need to be imposed. These transparency requirements involve having voters or other stakeholders verifying that all the processes of the election were done properly. Depending on what object is under verification, we distinguish between cast-as-intended verifiability, recorded-as-cast verifiability, counted-as-recorded verifiability and eligibility verifiability. Another classification can be done regarding who performs the verifications. In this case, we distinguish between individual verifiability, when only the entity responsible for the process (such as casting a vote) can perform the verification, and universal verifiability, when any entity can perform such verification.

Cast-as-intended verifiability refers to the process of verifying that the digital ballot cast by the voting device corresponds to the voting option chosen by the voter. As only the voter knows what voting option she chose, it is usually perceived that cast-as-intended verifiability belongs to individual verifiability.

Recorded-as-cast verifiability is concerned about verifying that the digital ballot which was cast by the voting device has been correctly stored by the electronic voting system. In this case, the verifiability requirement is only concerned about the digital ballot, not about the underlying chosen voting option. This verifiability property is also part of individual verifiability because only the voter knows that she cast a ballot and which one it is. However, the voter can delegate this verification to another party without compromising her privacy by telling such party which is her digital ballot.

We say that an electronic voting system has counted-as-recorded verifiability if it can be verified that the tallying process was done properly. Such verification is done with respect to the digital ballots stored by the system. In this case, anyone can potentially perform such verification as it does not involve the voting process. Therefore, this verifiability property is part of universal verifiability.

Finally, we define the property of being able to verify that all recorded votes correspond to valid voters as eligibility verifiability. This verification can also be done by any stakeholder of the election, if we assume that the list of valid voters is public. This means that eligibility verifiability is part of universal verifiability.

It is easy to think about an electronic voting scheme which satisfies all these verifiability requirements. Indeed, the most transparent electronic voting scheme is the one in which a digital ballot is just the voting option chosen by the voter together with some identity of the voter. After the voting period, all digital ballots could be published on-line and any entity could compute the result of the election. In this case, the voter could trivially verify that her digital ballot corresponds to her chosen voting option and that it was recorded by the system. Anyone could verify that the announced result of the election corresponds to the digital ballots registered by the system. Given a list of eligible voters, any entity could also verify that each digital ballot was created by an eligible voter. The electronic voting system would be completely transparent and it would satisfy all the verifiability properties!

However, it would not be private at all. Designing electronic voting systems which satisfy both privacy and verifiability requirements is a challenging task and an interesting area of research.

1.2.3 Validatability requirements

Even if an electronic voting scheme satisfies all privacy and verifiability requirements it does not mean that people will trust the electronic voting system. One of the facts that helps to build trust in the system is to make sure that the tallying process will not output invalid votes. In other words, the process of casting a vote should be designed so that the voting system will not accept digital ballots corresponding to invalid votes.

A digital ballot could correspond to an invalid vote due to many reasons. First, the digital ballot itself might be meaningless. The clearest case is the one in which the digital ballot is random nonsense data.

However, it might also happen that the digital ballot *looks like* a valid ballot but that it does not correspond to any meaningful voting option. Let me give an example: in a referendum, a voter might either vote "yes", "no" or "blank". We could imagine a scenario in which the digital ballot corresponds to a voting option which is neither of these three options. In this case, the vote would be invalid, but this might not be easy to infer from the ballot itself (think of a closed envelope: it does not reveal whether a vote is invalid).

Referendums are a concrete case of elections in which voters choose one out of three options. We could consider more complex elections in which the voter can choose t out of n candidates. In this case, if the voter chose the same candidate more than once then she would be choosing an invalid vote.

We could even imagine more complex elections in which the voter can make several selections and where there are several rules which state which are the valid combinations. In these complex elections, there could be many reasons for which a vote might be invalid - one for each combination of validity rules.

The difficulty of satisfying the validatability requirement ultimately depends on what types of invalid votes we can have in an election. Therefore, it is reasonable to make research on validatability properties of electronic voting schemes which correspond to elections with different voting rules.

1.3 Using cryptography to meet security requirements

Designing electronic voting systems which satisfy privacy, verifiability and validatability requirements at the same time seems an impossible quest. However, the use of cryptography has proven to be helpful in building systems that satisfy such requirements. A deep understanding of the cryptographic primitives is essential to build secure electronic systems.

1.3.1 Using encryption to achieve privacy

To start with, encryption schemes are cryptographic primitives which help protect data privacy. Encryption schemes come in two flavours: symmetric encryption, in which both the sender and the receiver of the information must share a key, and asymmetric encryption, in which the sender encrypts the information to be sent to the receiver using the receiver's public key, who uses her secret key to decrypt it.

In electronic voting, voters send their votes privately to the election authorities. However, it would be impractical for the election authorities to privately share a different key with every voter, so asymmetric encryption schemes are used. The electoral board has one public key, which any voter can use to encrypt her vote, and a private key, which is only known to the election board and is used to decrypt ballots.

There exist many asymmetric encryption schemes, with different security guarantees. It is important to know which security guarantee each encryption scheme provides since an incorrect choice of a secure encryption scheme might lead to an insecure voting scheme, if the security guarantees from the encrypted scheme are not the expected ones.

In addition, in some cases we will require that the encryption scheme has some special properties, usually not required for general data privacy. To protect the privacy of the voters, the election authorities apply some vote anonymization techniques before decrypting them. The most widely used techniques involve homomorphically aggregating the digital ballots or shuffling and re-encrypting them multiple times. If the encryption scheme has homomorphic properties then the digital ballots can be homomorphically aggregated to obtain an encryption of the result of the election, which can then be decrypted by the election authorities. On the other hand, homomorphic encryption schemes also allow us to re-encrypt ciphertexts without increasing their size, making those schemes ideal for shuffle and re-encryption anonymization techniques.

1.3.2 Using signature schemes to achieve integrity and authenticity

Another widely studied cryptographic primitive is signature schemes. A signature scheme allows a sender to produce a signature on some information by using a private key. By using the corresponding public key, any entity can verify such signature.

The signature guarantees not only that the signed information comes from the sender and not from some impostor but also that the signed information has not been tampered with. One application of digital signatures in electronic voting is to sign digital ballots: we don't want anyone to tamper with ballots and we usually want to be able to identify who cast a ballot to make sure that no entity is producing multiple ballots.

In addition, signature schemes can also be used in other parts of the process. Election authorities may sign election information such as the list of the candidates or the announced tally. In this thesis, signature schemes also have some uses for satisfying other electronic voting requirements such as vote validatability.

1.3.3 Using zero-knowledge proofs to achieve transparency

As mentioned above, it is essential that electronic voting systems are transparent. We want to be sure that the election authorities are following the protocol correctly and that they are producing the correct results, that voting devices are correctly producing their ballots and that voters do not create spoiled ballots. However, we want to do this without breaking privacy.

This is where zero-knowledge proofs come into play. A zero-knowledge proof allows a prover to convince a verifier that some statement is true without revealing anything else than the truth of the statement. These statements can be of the form "the tally is correctly computed from the ballots". In this case, zero-knowledge proofs allow election authorities to convince everyone that the statement is true without revealing the contents of any ballot.

Zero-knowledge proofs can be interactive, when the prover and the verifier send messages back and forth, or non-interactive, where the prover produces the proof without interaction with any verifier, and this proof can then be verified by any entity without interacting with the prover. In electronic voting the most used zero-knowledge proofs are the non-interactive ones, since we do not want the election authority to have to interact with every voter to convince them that the tally was correctly generated.

Recently, Groth and Sahai [GS12] introduced some powerful non-interactive zero-knowledge proofs which have been extensively used to build other cryptographic protocols. Since their introduction, Groth-Sahai proofs have received a lot of attention and are still an active research topic.

1.4 Objectives

The initial objective of the thesis was to understand which the main challenges that electronic voting systems face are and to build electronic voting protocols solving (some of) these challenges. While other researchers might focus on rather theoretical aspects of electronic voting, we were particularly interested in those challenges which have a significant impact on real-world electronic voting usage.

One example is those issues which impact voters' experience: we need to design them not for abstract voters, who will always follow any process flawlessly, but for real voters, who might not always be able to follow complex workflows. Building cast-as-intended verifiable electronic voting protocols is one of such challenges: it is the voter who needs to verify that her ballot contains her chosen voting option, so we should build protocols in which this verification is as simple as possible.

Similarly, we are interested on building mechanisms which help to build trust on electronic voting: a perfect electronic voting protocol will not be used in the real world if it works in a way which is not easily understood by its users. We have already mentioned the concept of validatability: a property that we introduce in this thesis that, while not being essential for the security of an electronic voting protocol, it is required if we want an electronic voting system to be trusted.

While we wanted to build electronic voting protocols solving real-world issues, we understood that to have *secure* electronic voting protocols we needed to design them with a detailed and formal treatment of their security properties. This meant understanding which is the state of the art in security requirements for electronic voting protocols and how they are formalized. This aspect of the research was paramount importance since we might need to modify the existing security definitions to accommodate any new property or functionality which our protocols might have.

To be able to build electronic voting protocols, it was essential to have a deep understanding of the cryptographic tools which are used to build them. As mentioned above, one of the most important of such tools is zero-knowledge proofs, so we decided to deepen our understanding of this area of cryptography.

1.5 Contributions and structure of this thesis

This thesis' contributions which can be split in two categories. On the one hand, there are contributions in cryptography and, in particular, in zero-knowledge proofs. The other contributions are in the field of electronic voting.

In Section 2 we give all the cryptographic background related to both our contributions in cryptography and electronic voting. This cryptographic background includes the assumptions that we work with, definitions of encryption schemes, signature schemes, zero-knowledge proofs and other cryptographic primitives.

The first contribution in the field of cryptography is given in Section 3. In this contribution, we give a framework to generalize cryptographic assumptions such as the Decisional Diffie Hellman assumption or the Decisional Linear assumption. This framework allows us to give new cryptographic assumptions which can be used to give more efficient instantiations of cryptographic protocols. One of the cryptographic protocols which can be instantiated with our new computational assumptions is Groth-Sahai proofs. The resulting instantiations are more efficient than previously known ones. In addition, we use our framework to build shorter Non-Interactive Zero-Knowledge proofs with security based on *traditional* assumptions for some concrete statements, such as equality of plaintexts in different ciphertexts.

The second contribution in the field of cryptography is given in Section 4. In this contribution, we expand Groth-Sahai proofs in several directions. We re-frame Groth-Sahai proofs as a Commit-and-Prove scheme, which allows us to reuse parts of the proofs in different related statements, therefore reducing the total size of those proofs. We also show how Groth-Sahai proofs can be made more efficient for some subclass of statements by introducing new techniques. Finally, we show how to reduce the prover's computation when she must produce proofs for multiple statements, even if the statements are completely unrelated.

In Section 5 we explain what is the state of the art in electronic voting research. We introduce which are the main types of electronic voting protocols and we give an overview of the typical security requirements.

Our first contribution in electronic voting is given in Section 6. In this section, we present a new security requirement for electronic voting protocols: vote validatability. This requirement states that it should be possible to publicly verify if a ballot is valid or invalid, therefore preventing the presence of previously undetected invalid votes when tallying the election. In addition to introducing the security requirement, which we formally define, we also give a general construction of an electronic voting protocol built from basic cryptographic primitives and we give an efficient instantiation of this electronic voting protocol.

We give our last contribution in Section 7. In this contribution, we show that it is possible to have cast-as-intended mechanisms which can be used by any entity to verify that a ballot contains the voting option chosen by the voter. This contrasts with previous proposals, in which only the voter of the cast vote could perform such verification. We call this property cast-as-intended verifiability. We formally define the universal cast-as-intended verifiability property, we give a generic construction of an electronic voting protocol satisfying this property and we show how to efficiently instantiate this generic construction. In addition, we present some possible implementations of electronic voting systems based on our protocol as a first step towards building usable electronic voting systems with universal cast-as-intended verifiability.

Finally, we present the conclusions of this thesis in Section 8.

Chapter 2

Cryptography background

In this section, we introduce some of the most important primitives in cryptography and, in particular, those that are most often used in electronic voting systems.

2.1 Some preliminaries

Before introducing the basic cryptographic primitives, we introduce some concepts and notation which we will use throughout the following sections.

Notation

We write $y = A(x; r)$ when the algorithm A, on input x and randomness r, outputs y. We write $y \leftarrow A(x)$ for the process of picking randomness r uniformly at random and setting $y = A(x; r)$. More generally, we write $y \leftarrow S$ for sampling y from the set S according to some probability distribution on S, using the uniform distribution as the default when nothing else is specified.

We write $a \leftarrow A; b \leftarrow B(a); \ldots$ for running the experiment where a is chosen from A, then b is chosen from B, which may depend on a, etc. This yields a probability distribution over the outputs, and we write $\Pr[a \leftarrow A; b \leftarrow B(a); \cdots : C(a, b, \ldots)]$ for the probability of the condition $C(a, b, \ldots)$ being satisfied after running the experiment.

Negligible and overwhelming functions

Given two functions $f, g : \mathbb{N} \to [0, 1]$ we write $f(k) \approx g(k)$ when $|f(k) - g(k)| = O(k^{-c})$ for every positive integer c. We say that f is negligible when $f(k) \approx 0$ and that it is overwhelming when $f(k) \approx 1$.

Distinguishing distributions

We say that two sequence of probability distributions $\{X_k\}_k$ and $\{Y_k\}_k$ are computationally indistinguishable if, for any probabilistic polynomial time (p.p.t.) adversary $\mathcal{A}$, its advantage defined as:

$$\mathbf{Adv}(k) := \Pr[\mathcal{A}(X_k) \text{ outputs } 1] - \Pr[\mathcal{A}(Y_k) \text{ outputs } 1]$$

is negligible.

On the other hand, two sequence of probability distributions $\{X_k\}_k$ and $\{Y_k\}_k$ are said to be identical if their support is the same and the same probability is assigned for each value in both probability distributions.

In some cases, the sequence of probability distributions will be implicitly parametrized by a security parameter λ. In this case, we might just say that the two probability distributions are indistinguishable but we will be referring to the indistinguishability of the sequence of probability distributions.

2.2 Groups and assumptions

Many cryptographic schemes are based on commutative groups and, in particular, on prime order groups. A group description $(p, \bar{\mathbb{G}}, \bar{g})$ consists of the group $\bar{\mathbb{G}}$ of prime order p and a generator of the group $\bar{g}$. These group descriptions are generated by a p.p.t. algorithm $\mathcal{G}$, which takes as input a security parameter 1^λ and outputs a group description $(p, \bar{\mathbb{G}}, \bar{g})$. We write group elements $\bar{x} \in \bar{\mathbb{G}}$ with a bar to distinguish them from elements in $\mathbb{Z}_p$ and we denote the neutral element in $\bar{\mathbb{G}}$ as $\bar{0}$. In addition, unless stated otherwise, we will write group operations with additive notation, this is, elements $\bar{x}, \bar{y} \in \bar{\mathbb{G}}$ are operated as $\bar{x} + \bar{y} = \bar{z} \in \bar{\mathbb{G}}$ and we will write $\bar{x} + \overset{(a)}{\cdots} + \bar{x} = a\bar{x}$ for $a \in \mathbb{Z}_p, \bar{x} \in \bar{\mathbb{G}}$. This notation deviates from standard practice but will greatly simplify our results and will make it possible to use linear algebra concepts such as vectors and matrices in a natural way. We stress that even though we are using additive notation it is hard to compute discrete logarithms in the groups.

There are groups for which it is assumed that some computational problems are hard. When cryptographic protocols are based on these groups, we reduce their security to the hardness of these problems. One of such computational assumptions is the Decisional Diffie Hellman (DDH) assumption.

Definition 2.2.1. *The DDH problem in $\bar{\mathbb{G}}$ is to distinguish between the two distributions $(\bar{g}, a\bar{g}, b\bar{g}, ab\bar{g})$ and $(\bar{g}, a\bar{g}, b\bar{g}, z\bar{g})$ where $a, b, z \leftarrow \mathbb{Z}_p$. We say that the DDH assumption holds relative to $\mathcal{G}$ if the DDH problem is computationally hard in $\bar{\mathbb{G}}$ for $(p, \bar{\mathbb{G}}, \bar{g}) \leftarrow \mathcal{G}$.*

Another mathematical structure with particular interest to cryptography is bilinear groups. A bilinear group consists of three prime order groups $\hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}$, an efficiently computable, non-degenerate bilinear map $e : \hat{\mathbb{G}} \times \check{\mathbb{H}} \to \mathbb{T}$, often referred to as a pairing, and generators $\hat{g}, \check{h}$ of $\hat{\mathbb{G}}, \check{\mathbb{H}}$ respectively. We write elements $\hat{x} \in \hat{\mathbb{G}}$ with a hat, elements $\check{y} \in \check{\mathbb{H}}$ with an inverted hat and elements $t_{\mathbb{T}} \in \mathbb{T}$ with the $\mathbb{T}$ subscript. We denote the neutral elements in the groups $\hat{\mathbb{G}}, \check{\mathbb{H}}$ and $\mathbb{T}$ with $\hat{0}, \check{0}$ and $0_{\mathbb{T}}$. In some cases, it will also be convenient to write the pairing e with multiplicative notation, defining $\hat{x}\check{y} = e(\hat{x}, \check{y})$.

As explained in [GPS08], one can separate known bilinear groups in three basic types:

type-I, in which $\hat{\mathbb{G}} = \check{\mathbb{H}} = \bar{\mathbb{G}}$ are the same group. In this case, we write elements $\bar{x} \in \bar{\mathbb{G}}$ with a bar.

type-II, in which $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$ are different but there exists an efficiently computable homomorphism $\phi : \check{\mathbb{H}} \to \hat{\mathbb{G}}$ which inverse ϕ^{-1} is not efficiently computable.

type-III, in which there are no efficiently computable homomorphisms between $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$.

Type-I bilinear groups are also known as symmetric bilinear groups, and their description $(p, \bar{\mathbb{G}}, \mathbb{T}, e, \bar{g})$ consists of the order of the groups p, the groups $\bar{\mathbb{G}}, \mathbb{T}$, the bilinear map e and a generator $\bar{g}$ of $\bar{\mathbb{G}}$. The DDH assumption can not hold in symmetric bilinear groups, since $e(a\bar{g}, b\bar{g}) = e(ab\bar{g}, \bar{g})$ can be used for distinguishing the two distributions.

This does not mean that symmetric bilinear groups are not useful for cryptography. Indeed, many constructions base their security on the Decisional Linear assumption (DLIN), which might hold even under the existence of the bilinear map e. In the following, let $\mathcal{SPG}$ be a p.p.t. algorithm which on input a security parameter 1^λ returns a symmetric bilinear group description $(p, \bar{\mathbb{G}}, \mathbb{T}, e, \bar{g})$.

Definition 2.2.2. *The DLIN problem in $\bar{\mathbb{G}}$ is to distinguish between the two distributions $(\bar{g}, a\bar{g}, b\bar{g}, ra\bar{g}, sb\bar{g}, (r+s)\bar{g})$ and $(\bar{g}, a\bar{g}, b\bar{g}, ra\bar{g}, sb\bar{g}, z\bar{g})$, where $a, b, r, s, z \leftarrow \mathbb{Z}_p$. We say that the DLIN assumption holds relative to $\mathcal{SPG}$ if the DLIN problem is hard in $\bar{\mathbb{G}}$ for $(p, \bar{\mathbb{G}}, \mathbb{T}, e, \bar{g}) \leftarrow \mathcal{SPG}$.*

By default, when talking about asymmetric bilinear groups we will be referring to type-III bilinear groups. We denote algorithms which generate asymmetric bilinear groups as $\mathcal{PG}$. An algorithm $\mathcal{PG}$ takes as input a security parameter 1^λ and returns a group description $(p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h})$, where $\hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}$ are groups of prime order p, $\hat{g}$ and $\check{h}$ are generators of $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$ respectively, and e is an efficiently computable, non-degenerate bilinear map $e : \hat{\mathbb{G}} \times \check{\mathbb{H}} \to \mathbb{T}$.

In asymmetric bilinear groups the DDH problem might be hard in both $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$, as the bilinear map e does not help to solve the DDH problem.

Definition 2.2.3. *We say that the Symmetric eXternal Diffie-Hellman (SXDH) assumption holds relative to $\mathcal{PG}$ if the DDH problem is computationally hard in both $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$ for $(p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathcal{PG}$.*

Finally, note that in type-II bilinear groups we might be able to assume that the DDH problem is hard in $\hat{\mathbb{G}}$ and the DLIN problem is hard in $\check{\mathbb{H}}$. The homomorphism ϕ and the bilinear map e can be used to trivially solve the DDH problem in $\check{\mathbb{H}}$ but not in $\hat{\mathbb{G}}$.

Depending on the bilinear group being used, cryptographic schemes are based on different assumptions. In particular, each type of bilinear group leads to different constructions, with type-I bilinear groups leading to the most efficient schemes in terms of group elements. However, as detailed in [GPS08], known constructions of the different bilinear group types have important trade-offs in properties such as of group sizes (for the same security parameter), group operation efficiency or pairing efficiency.

2.3 The Random Oracle Model and Hash Functions

A random oracle [BR93] is an algorithm that takes as input a value from some domain, processes it, and returns a value from some range. A random oracle processes its inputs as follows: on an input which has not been received before, it samples a random element from its range, stores the relation between the input and this element and returns the sampled element. For inputs which the random oracle has already seen, it looks at the corresponding sampled element and returns it. We say that a cryptographic scheme is secure in the Random Oracle Model (ROM) if it uses random oracles and satisfies the appropriate security definitions.

Random oracles can not be implemented as a cryptographic scheme: two entities using the same random oracle on the same input should obtain the same result, but this is impossible to implement due to its random behavior and to due the state that needs to be kept in the random oracle. Therefore, when instantiating cryptographic primitives we usually substitute the random oracle by a *hash function*.

Hash functions are one of the basic cryptographic primitives. A hash function H is a function which takes as input bit strings of arbitrary size and compresses them into a constant-size output. There are three basic security requirements for hash functions:

Pre-image resistance: given a hash value h it should be difficult to find a message m such that $H(m) = h$.

Second pre-image resistance: given a message m_1 it should be difficult to find a different message m_2 such that $H(m_1) = H(m_2)$.

Collision resistance: it should be difficult to find two different messages m_1, m_2 such that $H(m_1) = H(m_2)$.

There are many efficient constructions of hash functions, such as the standardized SHA-256 [Dan13]. It is assumed that these hash functions are secure since no attacks have been found against them.

The outputs of hash functions usually look like random values and, for this reason, they are used to instantiate random oracles. However, the security which results from substituting a random oracle by a hash function is heuristic. There are protocols designed so that they are secure in the Random Oracle Model but, when random oracles are instantiated by any cryptographic hash function, they can not be secure [CGH04].

2.4 Encryption schemes

The first primitive which comes to mind when one talks about cryptography is encryption. Encryption is about transforming a message that anyone can read into a ciphertext which reveals no information about the message. Obviously, encryption is also about reversing this process: turning the ciphertext back into a meaningful message. To perform each of these operations, called encrypting and decrypting, a key is used.

Encryption comes in two flavors: symmetric and asymmetric. In symmetric encryption, the processes of encrypting and decrypting the message use the same key. Anyone who can encrypt a message can also decrypt any ciphertext encrypted using the same key. On the other hand, in asymmetric encryption, the encrypting process uses a key, referred to as the *public key*, and the decryption process uses another key, referred to as the *private key*. Usually, the public key is published (this is why it receives this name), so that anyone can encrypt any message using this public key. The private key is only known by the intended receiver of these messages, so that only she can decrypt the ciphertexts and retrieve the original messages.

Electronic voting usually happens in an asymmetric scenario. On the one hand, we have multiple voters who send their ballots to some election authority. On the other hand, we have the election authority, who tallies the election from the votes contained in the ballots. Naturally, most of the electronic voting schemes are built on asymmetric encryption schemes.

The syntactical definition of an encryption scheme is the following one.

$\mathsf{Setup}(1^\lambda)$: on input a security parameter 1^λ outputs a setup gk. This implicitly defines (possibly setup-dependant) spaces: the message space $\mathfrak{M}_{gk}$, the ciphertext space $\mathfrak{C}_{gk}$ and the encryption randomness space $\mathfrak{R}_{gk}$.

$\mathsf{KeyGen}(gk)$: on input a setup gk it generates and outputs a public key pk and a secret key sk.

$\mathsf{Enc}(gk, pk, m; r)$: on input a public key pk and a message $m \in \mathfrak{M}_{gk}$, outputs a ciphertext $c \in \mathfrak{C}_{gk}$ using randomness $r \in \mathfrak{R}_{gk}$. In some cases, we might omit the randomness used and just write $\mathsf{Enc}(gk, pk, m)$.

$\mathsf{Dec}(gk, sk, c)$: on input a secret key sk and a ciphertext $c \in \mathfrak{C}_{gk}$ outputs a message $\tilde{m} \in \mathfrak{M}_{gk}$ or aborts outputting an error symbol $\perp$.

We say that an encryption scheme is **correct** if, for all setups generated by the algorithm Setup, all public/private key pairs generated by the algorithm KeyGen and for all messages in the message space $m \in \mathfrak{M}_{gk}$ it is satisfied that:

$$\mathsf{Dec}(gk, sk, \mathsf{Enc}(gk, pk, m)) = m.$$

IND-CPA security

The basic notion of security for an encryption scheme is Indistinguishability of ciphertexts under a Chosen Plaintext Attack (IND-CPA) [MRS88]. This notion considers a very weak adversarial goal: to guess which of two adversarially generated messages is encrypted in a ciphertext, with probability better than a random coin flip. As for the access to the encryption scheme, it considers basic access: the adversary knows the public key of the encryption key pair. This security notion is formalized via the following experiment between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

1. First, the challenger generates a setup $gk = \mathsf{Setup}(1^\lambda)$ and a key pair $(sk, pk) = \mathsf{KeyGen}(gk)$. It keeps sk for itself and forwards (gk, pk) to the adversary.
2. Then, the adversary submits two messages (m_0, m_1) to the challenger.
3. The challenger draws a random bit $b \leftarrow \{0, 1\}$. It then computes $c = \mathsf{Enc}(gk, pk, m_b)$ and sends the ciphertext c to the adversary.
4. Finally, the adversary submits a bit $\tilde{b}$.

We say that an encryption scheme is IND-CPA secure if, for all p.p.t. adversaries, the advantage of the adversary defined as follows is negligible:

$$\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}(\lambda) := |\Pr[\tilde{b} = b] - 1/2|$$

NM-CPA security

Let's consider encryption in the scenario of electronic voting. As we will see, ballots cast by voters are often public since they do not reveal any information about the chosen voting options. A voter can then copy someone else's ballot and cast it as if it was her own ballot. This is not a desired behavior since it might be useful for coercing voters (asking them to submit the same ballot that the coercer submitted) or breaking privacy (if only a few voters participate in an election and many of them are corrupt they could copy a honest voter's ballot and infer information from the tally).

Simply copying a ballot can be easily detected by comparing each received ballot with all previously received ballots. However, a voter might try to copy another voter's ballot and exploit properties of the encryption protocol to transform the copied ballot into a ballot with the same contents but which looks completely different from the original one.

We therefore need that the encryption scheme does not allow for such kind of tampering with ciphertexts. This is captured in the security definition of Non-Malleability of ciphertexts under a Chosen Plaintext Attack (NM-CPA) [BDPR98]. The access of the adversary is the same than in the IND-CPA definition: it only has access to the public key of the encryption key pair. However, its goal is different: the adversary now tries to tamper with the ciphertext to convert it into an encryption of a related message.

The formal definition, which we now present, states this goal in the following way: the adversary receives a ciphertext from a challenger and tries to output a ciphertext which plaintext satisfies some relation with the original plaintext. However, he must do better (in terms of probability) than outputting a ciphertext which plaintext satisfies the same relation with a random plaintext. Instead of specifying the original plaintext, though, the adversary specifies a distribution of plaintexts from which the original and the random plaintexts are drawn. This security notion is formalized via the following experiment between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

1. First, the challenger generates a setup $gk = \mathsf{Setup}(1^\lambda)$ and a key pair $(sk, pk) = \mathsf{KeyGen}(gk)$. It keeps sk for itself and forwards (gk, pk) to the adversary.
2. Then, the adversary outputs a distribution of messages $\mathcal{M}_\mathcal{A}$ i.e., a subset of the message space $\mathfrak{M}_{gk}$ where each message has attached a non-zero probability (being the sum of all probabilities 1).
3. The challenger samples two messages $m_0, m_1 \leftarrow \mathcal{M}_\mathcal{A}$ and a random bit $b \leftarrow \{0, 1\}$. It then computes $c = \mathsf{Enc}(gk, pk, m_0)$, sends the ciphertext c to the adversary and keeps m_0, m_1 for himself.
4. The adversary submits a ciphertext c^*, different than c, and a relation R.
5. Finally, the challenger decrypts the ciphertext c^* obtaining m^*. If $m^* =\perp$ then the experiment outputs a bit $\alpha = 0$. Otherwise, the experiment outputs $\alpha = 1$ if the relation R holds between m^* and m_b and $\alpha = 0$ otherwise.

We say that an encryption scheme is NM-CPA secure if, for all p.p.t. adversaries, the advantage of the adversary defined as follows is negligible:

$$\mathbf{Adv}^{\mathsf{NM\text{-}CPA}}(\lambda) := |\Pr[\alpha = 1|b = 1] - \Pr[\alpha = 1|b = 0]|$$

IND-CCA security

Another well-known security property of encryption schemes is the property of Indistinguishability against Chosen Ciphertext Attacks (IND-CCA). This security properties comes in two flavors, known as IND-CCA1 [NY90] and IND-CCA2 [RS91]. We will only consider the IND-CCA2 notion and we will write it as IND-CCA.

This security notion models an adversary with the same goal as the IND-CPA adversary, which is to distinguish between the encryption of two chosen messages. However, the IND-CCA adversary has a much stronger access: it can get decryptions for any ciphertext but the target ciphertext, in addition to having the public key of the encryption key pair. This access models the fact that an adversary might trick the receiver of ciphertexts into revealing some information about the plaintext. For example, if in some protocol the decryption process aborts because the ciphertext is malformed and the receiver returns some error message, the adversary might be gaining useful information. Similarly, if the receiver decrypts a message which she does not expect, she might notify the adversary. As the access of the IND-CCA adversary is much stronger than the access of an IND-CPA adversary, an IND-CCA secure encryption scheme is also IND-CPA secure. The security notion is formalized via the following experiment between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

1. First, the challenger generates a setup $gk = \mathsf{Setup}(1^\lambda)$ and a key pair $(sk, pk) = \mathsf{KeyGen}(gk)$. It keeps sk for itself and forwards (gk, pk) to the adversary.
2. The adversary might then send ciphertexts c to the challenger, who decrypts them and returns the result to the adversary. This can be done a number of times polynomial on the security parameter.
3. Then, the adversary submits two messages (m_0, m_1) to the challenger.
4. The challenger draws a random bit $b \leftarrow \{0, 1\}$. It then computes $c^* = \mathsf{Enc}(gk, pk, m_b)$ and sends the ciphertext c^* to the adversary.
5. The adversary might then send more ciphertexts $c \neq c^*$ to the challenger, who decrypts them returning the result to the adversary. This can be done a polynomial number of times.
6. Finally, the adversary submits a bit $\tilde{b}$.

We say that an encryption scheme is IND-CCA secure if, for all p.p.t. adversaries, the advantage of the adversary defined as follows is negligible:

$$\mathbf{Adv}^{\mathsf{IND\text{-}CCA}}(\lambda) := |\Pr[\tilde{b} = b] - 1/2|$$

Weakening the access of the adversary, we can give other privacy definitions. An interesting one is known as IND-1-CPA. The goal of an IND-1-CPA adversary is once again to distinguish which of two chosen messages is encrypted in a ciphertext. However, the access of this adversary stands between the access of an IND-CPA adversary and an IND-CCA adversary: it has access to the public key and, after receiving the challenge ciphertext, it can ask once for a decryption of multiple ciphertexts. Note that this is different from the IND-CCA adversary access not only because it can not get decryptions before seeing the challenge ciphertext but also because he must send all the ciphertexts to be decrypted at once, thus not allowing any adaptive strategy in which each query depends on the previous one. The security notion is formalized via the following experiment between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

1. First, the challenger generates a setup $gk = \mathsf{Setup}(1^\lambda)$ and a key pair $(sk, pk) = \mathsf{KeyGen}(gk)$. It keeps sk for itself and forwards (gk, pk) to the adversary.
2. Then, the adversary submits two messages (m_0, m_1) to the challenger.
3. The challenger draws a random bit $b \leftarrow \{0, 1\}$. It then computes $c^* = \mathsf{Enc}(gk, pk, m_b)$ and sends the ciphertext c^* to the adversary.
4. The adversary might then send a vector of ciphertexts $(c_1, \ldots, c_n)$ to the challenger, who decrypts them returning the result to the adversary. None of the ciphertexts c_i can be equal to c^*.
5. Finally, the adversary submits a bit $\tilde{b}$.

We say that an encryption scheme is IND-1-CPA secure if, for all p.p.t. adversaries, the advantage of the adversary defined as follows is negligible:

$$\mathbf{Adv}^{\mathsf{IND\text{-}1\text{-}CPA}}(\lambda) := |\Pr[\tilde{b} = b] - 1/2|$$

An interesting result is given in [BS99], where they prove the following lemma:

Lemma 2.4.1. *NM-CPA security is equivalent to IND-1-CPA security.*

This lemma is interesting for two reasons. First, it relates the notion of NM-CPA security with the notions of IND-CPA security and IND-CCA security. A priori, relating these notions would be difficult since assessing whether the IND goal or the NM goal is stronger does not seem easy. However, due to this result we know that IND-CCA security is stronger than NM-CPA security, which is in turn stronger than IND-CPA security. On the other hand, working with the notion of NM-CPA security is difficult since it involves proving things for adversarially chosen relations, whereas working with IND-1-CPA security is considerably easier.

Homomorphic encryption

An encryption scheme works in a rather simple way: encrypting a message yields a ciphertext, which can be decrypted back to a message. Having a ciphertext without knowing the secret key is meaningless. However, in some scenarios we want to be able to perform operations on ciphertexts.

An example is encrypted search: a user submits a ciphertext to a search engine, which returns some encrypted search result based on the underlying plaintext, without ever decrypting the user's ciphertext. The primitive that allows us to do these sort of operations is known as *fully homomorphic encryption* (FHE). FHE was introduced in 2009 [Gen09] and is a very active area of research in cryptography. Despite recent advances in FHE its underwhelming performance still limits its real applications.

A related primitive is known as partially homomorphic encryption, often just referred as homomorphic encryption. If a FHE allows us to compute *any* operation on a ciphertext (which reduces to computing bit addition, multiplication and negation), homomorphic encryption only allows us to compute the group operation of the group the encryption scheme is built on. One example of a homomorphic encryption scheme is the Pailler encryption scheme [Pai99], in which the product of two ciphertext yields a ciphertext which is the encryption of the sum of the two original messages.

Homomorphic encryption turns out to be really useful for electronic voting. However, it is easily seen that a homomorphic encryption scheme can not be NM-CPA secure. The reason is that, in those schemes, ciphertexts are malleable. For example, it is easy to turn a ciphertext encrypting a plaintext m into a ciphertext encrypting a plaintext $2m$. For this reason, a homomorphic encryption scheme can not be IND-CCA secure neither, since it's a security property stronger than NM-CPA. However, there are NM-CPA encryption schemes which ciphertexts can be transformed into homomorphically aggregable ciphertexts.

2.4.1 Concrete encryption schemes

One of the most well-known encryption schemes is the ElGamal encryption scheme [Gam84]. The scheme is based on a group $\bar{\mathbb{G}}$ of prime order p and generator $\bar{g}$. Its message space is $\bar{\mathbb{G}}$, its randomness space is $\mathbb{Z}_p$ and its ciphertext space is $\bar{\mathbb{G}} \times \bar{\mathbb{G}}$. The scheme works as follows.

$\mathsf{Setup}(1^\lambda)$: on input a security parameter 1^λ outputs a group description $gk = (p, \bar{\mathbb{G}}, \bar{g}) \leftarrow \mathcal{G}(1^\lambda)$.

$\mathsf{KeyGen}(gk)$: on input a setup $gk = (p, \bar{\mathbb{G}}, \bar{g})$, it samples a random scalar $x \in \mathbb{Z}_p$ and computes $\bar{h} = x\bar{g}$. The secret key is x and the public key is $\bar{h}$.

$\mathsf{Enc}(gk, pk, m)$: on input a setup $gk = (p, \bar{\mathbb{G}}, \bar{g})$, a public key $pk = \bar{h}$ and a message $m = \bar{m} \in \bar{\mathbb{G}}$, samples a random $r \in \mathbb{Z}_p$ and outputs a ciphertext $c = (\bar{c}_1, \bar{c}_2) = (r\bar{g}, r\bar{h} + m)$.

$\mathsf{Dec}(gk, sk, c)$: on input a setup $gk = (p, \bar{\mathbb{G}}, \bar{g})$, a secret key $sk = x \in \mathbb{Z}_p$ and a ciphertext $c = (\bar{c}_1, \bar{c}_2) \in \bar{\mathbb{G}} \times \bar{\mathbb{G}}$ outputs a message $\bar{m}' = \bar{c}_2 - x\bar{c}_1$.

It is straightforward to see that the scheme is correct. In addition, the scheme is homomorphic:

$$\mathsf{Enc}(gk, pk, m_1) + \mathsf{Enc}(gk, pk, m_2) = \mathsf{Enc}(gk, pk, m_1 + m_2),$$

where the addition in $\bar{\mathbb{G}} \times \bar{\mathbb{G}}$ is performed element-wise. In [TY98] it is shown that the ElGamal encryption scheme is $\mathsf{IND\text{-}CPA}$ secure if the Decisional Diffie-Hellman assumption holds relative to $\mathcal{G}$.

Due to the scheme being homomorphic, it can not be $\mathsf{NM\text{-}CPA}$ nor $\mathsf{IND\text{-}CCA}$. In [TY98] the following variant of the scheme is defined, where $H : \bar{\mathbb{G}}^3 \to \mathbb{Z}_p$ is a random oracle (instantiated by a hash function):

$\mathsf{Setup}(1^\lambda)$: on input a security parameter 1^λ outputs a group description $gk = (p, \bar{\mathbb{G}}, \bar{g}) \leftarrow \mathcal{G}(1^\lambda)$.

$\mathsf{KeyGen}(gk)$: on input a setup $gk = (p, \bar{\mathbb{G}}, \bar{g})$, it samples a random scalar $x \in \mathbb{Z}_p$ and computes $\bar{h} = x\bar{g}$. The secret key is x and the public key is $\bar{h}$.

$\mathsf{Enc}(gk, pk, m)$: on input a setup $gk = (p, \bar{\mathbb{G}}, \bar{g})$, a public key $pk = \bar{h}$ and a message $m = \bar{m} \in \bar{\mathbb{G}}$, samples two random scalars $r, s \in \mathbb{Z}_p$. It then computes and outputs the ciphertext $c = (\bar{c}_1, \bar{c}_2, c_3, c_4) = (r\bar{g}, r\bar{h} + \bar{m}, H(r\bar{g}, r\bar{h} + \bar{m}, s\bar{g}), s + H(r\bar{g}, r\bar{h} + \bar{m}, s\bar{g})r \mod p)$.

$\mathsf{Dec}(gk, sk, c)$: on input a setup $gk = (p, \bar{\mathbb{G}}, \bar{g})$, a secret key x and a ciphertext $c = (\bar{c}_1, \bar{c}_2, c_3, c_4) \in \bar{\mathbb{G}}^2 \times \mathbb{Z}_p^2$ outputs a message $\bar{m}' = \bar{c}_2 - x\bar{c}_1$ if $H(\bar{c}_1, \bar{c}_2, c_4\bar{g} - c_3\bar{c}_1) = c_3$ and $\perp$ otherwise.

This scheme, known as the Signed ElGamal scheme, is an extension of the ElGamal encryption scheme in which the encrypt algorithm computes some extra elements which are then used by the decrypt algorithm to verify that a ciphertext is correctly computed. Intuitively, the random oracle prevents anyone from tampering with the ciphertext. The Signed ElGamal encryption scheme is $\mathsf{NM\text{-}CPA}$ secure in the Random Oracle Model [BPW12a] if the Decisional Diffie-Hellman assumption holds relative to $\mathcal{G}$.

There is yet another variant of the ElGamal encryption scheme, known as the Cramer-Shoup encryption scheme [CS98]. This scheme can be shown to be $\mathsf{IND\text{-}CCA}$ secure. While the Cramer-Shoup encryption scheme uses hash functions, they are not used to model any random oracle and, therefore, the security of the Cramer-Shoup encryption scheme is not based on the Random Oracle Model.

In our thesis, we are particularly interested on the ElGamal encryption scheme and the Signed ElGamal encryption scheme due to the homomorphic properties of the first one and the $\mathsf{NM\text{-}CPA}$ security of the second one. One interesting fact is that, given a Signed ElGamal ciphertext, one can first check whether $H(\bar{c}_1, \bar{c}_2, c_4\bar{g} - c_3\bar{c}_1) = c_3$ is satisfied and then strip c_3 and c_4 from the ciphertext, obtaining an ElGamal ciphertext. This new ciphertext can be homomorphically operated with other ciphertexts and, in some cases, protocol security is guaranteed if the previous verification equation holds for all the original ciphertexts.

2.5 Signature schemes

Cryptography is not only about encrypting and decrypting messages. Another of the most used cryptographic primitives is digital signatures. A digital signature is like a handwritten signature: given a message, an entity can sign it so that anyone else can verify that the message was indeed signed by such entity. This is achieved by using a pair of keys: a private key for signing messages and a public key for verifying signatures.

The syntactical definition of a signature scheme is the following one. A signature scheme is parametrized by a (possibly setup-dependant) message space $\mathfrak{M}_{gk}$ and a setup-dependant signature space Σ_{gk}. The message space states which are the messages that can be signed and the signature space determines the format of a signature. A signature scheme consists of four (possibly probabilistic) algorithms:

$\mathsf{Setup}(1^\lambda)$: on input a security parameter 1^λ outputs a setup gk. This implicitly defines (possibly setup-dependant) spaces: the message space $\mathfrak{M}_{gk}$ and the signature space Σ_{gk}.

$\mathsf{KeyGen}(gk)$: on input a setup gk it generates and outputs a public key pk and a secret key sk.

$\mathsf{Sign}(gk, sk, m)$: on input a setup gk, a private key sk and a message $m \in \mathfrak{M}_{gk}$, outputs a signature $\sigma \in \Sigma_{gk}$.

$\mathsf{Verify}(gk, pk, m, \sigma)$: on input a setup gk, a public key pk, a message $m \in \mathfrak{M}_{gk}$ and a signature $\sigma \in \Sigma_{gk}$ outputs 1 for success or 0 for failure.

As with encryption schemes, we first require a basic property which defines the functionality of a signature scheme. We say that a signature scheme is **correct** if, for all setups generated by the Setup algorithm, all public/private key pairs generated using the algorithm KeyGen and for all messages in the message space $m \in \mathfrak{M}_{gk}$ it is satisfied that $\mathsf{Verify}(gk, pk, m, \mathsf{Sign}(gk, sk, m)) = 1$.

EUF-CMA security

The most widely used security notion for signature schemes is Existential Unforgeability under a Chosen Message Attack ($\mathsf{EUF\text{-}CMA}$) [GMR88]. The security notion is formalized via the following experiment between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

1. First, the challenger generates a setup $gk = \mathsf{Setup}(1^\lambda)$ and a key pair $(sk, pk) = \mathsf{KeyGen}(gk)$. It keeps sk for itself and forwards (gk, pk) to the adversary.

2. Then, the adversary sends messages m to the challenger, who computes signatures σ on such messages and sends them back to the adversary. The adversary can make a number of such queries polynomial on the security parameter.

3. Finally, the adversary outputs a pair (m^*, σ^*). The output of the experiment is a bit $\beta = 1$ if m^* has not been queried to the challenger for a signature and $\mathsf{Verify}(gk, pk, m^*, \sigma^*)$=1 and is 0 otherwise.

We say that an encryption scheme is $\mathsf{EUF\text{-}CMA}$ secure if, for all p.p.t. adversaries, the advantage of the adversary defined as follows is negligible:

$$\mathbf{Adv}^{\mathsf{EUF\text{-}CMA}}(\lambda) := \Pr[\beta = 1]$$

Other security properties

There are some security properties which are related to $\mathsf{EUF\text{-}CMA}$ security which are worth considering. First, a digital signature on a message might not be deterministic. A weaker goal than existential unforgeability (resulting in a stronger definition) is the goal of producing a signature on a message even if the adversary has already seen a different signatures on the same message. This notion is known as strong Existential Unforgeability against Chosen Message Attacks ($\mathsf{sEUF\text{-}CMA}$) [ADR02] and is defined by the same experiment than $\mathsf{EUF\text{-}CMA}$ but the output of the experiment is also 1 if m^* has been sent to the challenger as a signature query and σ^* has not been received from the challenger as the query reply.

If the adversary is only allowed to submit signature queries before seeing the public key of the signature key pair, we say that the signature is weakly Existentially Unforgeable under a Chosen Message Attack ($\mathsf{w\text{-}EUF\text{-}CMA}$) [BB08]. The security notion is formalized via the following experiment between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

1. First, the adversary $\mathcal{A}$ sends a tuple of messages $(m_1, \ldots, m_n)$ to the challenger.

2. After receiving the messages, $\mathcal{C}$ generates a setup $gk = \mathsf{Setup}(1^\lambda)$ and a key pair $(sk, pk) = \mathsf{KeyGen}(gk)$. It keeps sk for itself and forwards (gk, pk) to the adversary. It also computes signatures on the messages received from the adversary and sends such signatures back to the adversary.

3. Finally, the adversary outputs a pair (m^*, σ^*). The output of the experiment is a bit $\beta = 1$ if m^* has not been queried to the challenger for a signature and $\mathsf{Verify}(gk, pk, m^*, \sigma^*)$=1 and is 0 otherwise.

As usual, we say that an encryption scheme is wEUF-CMA secure if, for all p.p.t. adversaries, the advantage of the adversary defined as follows is negligible:

$$\mathbf{Adv}^{\text{w-EUF-CMA}}(\lambda) := \Pr[\beta = 1]$$

Even though w-EUF-CMA security is weaker than EUF-CMA security (due to the access of the adversary being weaker), this security notion has some applications. For instance, in electronic voting we might have the election authorities produce signatures on the lists of eligible candidates before the election takes place and destroy the private signing key after that. In this case, the adversary gets the signatures on the messages when receiving the public signing key. Actually, the access of an adversary is even weaker, as it might only perform known message attacks.

2.5.1 Structure-Preserving Signatures

A particular type of interesting signature schemes are Structure Preserving Signature. Structure-preserving signatures are defined over bilinear groups and have the property that messages and signatures consist of group elements and the verification of signatures consists on evaluating the so-called pairing product equations: equations in $\mathbb{T}$ in which each element of $\mathbb{T}$ is expressed as the pairing of elements of $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$. This allows us to combine structure-preserving signatures and other cryptographic primitives in an algebraic way.

A particularly efficient Structure-Preserving Signature scheme is the one given in [AGOT14], built on a type-III bilinear groups $(p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h})$.

$\mathsf{Setup}(1^\lambda)$: on input a security parameter 1^λ outputs a setup $gk = (p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}, \hat{f})$, which consists of a group description $(p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathcal{PG}$ and a random element $\hat{f} \leftarrow \hat{\mathbb{G}}$.

$\mathsf{KeyGen}(gk)$: on input a setup $gk = (p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}, \hat{f})$ it samples a random scalar $x \in \mathbb{Z}_p$ and computes $\check{v} = x\check{h}$. The secret key is x and the public key is $\check{v}$.

$\mathsf{Sign}(gk, sk, m)$: on input a setup $gk = (p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}, \hat{f})$ a private key $sk = x$ and a message $m = \hat{m} \in \hat{\mathbb{G}}$ it samples $r \in \mathbb{Z}_p^*$ and outputs the signature $\sigma = (\check{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_3) = (r\check{h}, \frac{x}{r}\hat{m} + \frac{x}{r}\hat{f}, \frac{x}{r}\hat{\sigma}_2 + \frac{1}{r}\hat{g}) \in \check{\mathbb{H}} \times \hat{\mathbb{G}}^2$.

$\mathsf{Verify}(gk, pk, m, \sigma)$: on input a setup $gk = (p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}, \hat{f})$, a public key $pk = \check{v}$, a message $m = \hat{m} \in \hat{\mathbb{G}}$ and a signature $\sigma = (\check{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_3)$ it outputs 1 if $e(\hat{\sigma}_2, \check{\sigma}_1) = e(\hat{m}, \check{v}) + e(\hat{f}, \check{h})$ and $e(\hat{\sigma}_3, \check{\sigma}_1) = e(\hat{\sigma}_2, \check{v}) + e(\hat{g}, \check{h})$ hold and 0 otherwise.

In [AGOT14], the authors show that this structure-preserving signature scheme is sEUF-CMA secure in the generic group model [Sho97], a security model which assumes that only a random encoding of the bilinear group is accessible by all entities.

2.6 Other cryptographic primitives

In this section, we present other cryptographic primitives. Even though they are not central to this thesis we use them as building blocks of some of our protocols.

One-way functions

Just like an encryption scheme, one-way functions transform some value into another value so that it is difficult to recover the original value. However, a one-way function is simpler than an encryption scheme: it is deterministic, it does not rely on any key and there is no “decryption” algorithm. As its name states, a one-way function is a function which is easy to evaluate but evaluating its inverse is hard.

More formally, a function $F : X \to Y$ between two finite sets is said to be one-way if the following two properties are satisfied, where $\lambda = \log |X|$:

- For each $x \in X$, $F(x)$ can be computed in time polynomial in λ.
- For any adversary $\mathcal{A}$ running in time polynomial in λ, and for $x \in X$ chosen with the uniform distribution, the probability that $\mathcal{A}$, on input $F(x)$, outputs x is negligible.

One-way functions are at the very core of cryptography. Among others, signature schemes can be built on one-way functions [Lam79]. In this thesis, we use one-way functions as one of the building blocks of our voting protocols.

One of the most well-known one-way function is the exponentiation function in a group where the discrete logarithm assumption [DH76] holds. Given such a group $(p, \bar{\mathbb{G}}, \bar{g}) \leftarrow \mathcal{G}(1^\lambda)$ and an element $\bar{f} \in \bar{\mathbb{G}}$, the function $F_{\bar{f}} : \mathbb{Z}_p \to \bar{\mathbb{G}}$ defined as $F_{\bar{f}}(x) = x\bar{f}$ is one-way. Actually, the discrete logarithm assumption and the one-wayness of $F_{\bar{f}}$ are exactly the same thing. The exponentiation function and the discrete logarithm assumption receive their name because traditionally the group $\bar{\mathbb{G}}$ is written in multiplicative notation.

An interesting subclass of one-way functions are homomorphic one-way functions. A one-way function is homomorphic if X and Y are groups, which we write in additive notation, and $F(x + y) = F(x) + F(y)$ for $x, y \in X$. The exponentiation function is homomorphic.

Pseudo-random permutations

A Pseudo-Random Permutation (PRP) family is a family of efficient bijective functions $F_{(\cdot)} : X \to X$ parametrized by a key $k \in K_{PRP}$.

The pseudo-random property of a PRP family states that it is difficult to distinguish the outputs of a function F_k for a random key $k \in K_{PRP}$ from those of a random permutation from X to X.

In other words, an adversary being able to query the function as a black-box on inputs of its choice should not be able to tell whether it is interacting with a random permutation or a pseudo-random permutation. In our voting protocols, we will use pseudo-random permutations to detect when the pre-image of the function takes the same value more than once, without giving any other information about the pre-image.

In [MRV99] it is shown RSA function family $F_{(\cdot)} : \mathbb{Z}_N \rightarrow \mathbb{Z}_N$ defined as $F_e(x) = x^e \mod N$ for $N = pq$ a product of primes and $(e, \phi(N)) = 1$ is a PRP family. Even though we are not going to use this particular PRP in this thesis, it is a useful example to understand this concept.

Secret sharing schemes

Secret sharing is a method to distribute a secret in several pieces or shares. The secret is distributed in such a way that it can only be reconstructed from an authorized subset of shares. Even more: a non-authorized subset of shares does not leak any information about the secret.

We now give the syntactical definition of a secret sharing scheme. For sake of simplicity, assume that an authorized subset of shares is any set of shares which contains at least t different shares. We call such schemes threshold secret sharing schemes. A secret sharing scheme is defined by two (possibly probabilistic) algorithms:

- $\mathsf{Split}(s, n)$: on input a secret s and an integer n it outputs n shares of the secret $(s_1, \ldots, s_n)$.
- $\mathsf{Reconstruct}(s_{i_1}, \ldots, s_{i_t})$: on input t shares $(s_{i_1}, \ldots, s_{i_t})$ outputs the reconstructed secret $\tilde{s}$.

The requirements of a secret sharing scheme are *correctness* and *perfect privacy*. Correctness states that, if t shares are used to reconstruct the secret then the original secret must be recovered. Perfect privacy states that less than t shares do not leak any information about the secret.

The most well-known secret sharing scheme is Shamir's Secret Sharing scheme [Sha79], in which a random polynomial over $\mathbb{Z}_p$ is used to split the secret into shares. Each share is defined to be the evaluation of the polynomial on a public value. One can control the threshold t by changing the degree of the polynomial: if the secret is shared by using a polynomial of degree $t-1$ then t shares are needed to reconstruct the secret whereas $t-1$ shares do not give any information about the secret.

2.7 Zero-Knowledge proofs

One of the cryptographic primitives with many applications to electronic voting is zero-knowledge proofs. The goal of this primitive is to prove that some statement is true without revealing any information other than the truthfulness of the statement. For example, we might want to prove that some group element is in the subgroup generated by another group element while hiding their discrete logarithm.

In electronic voting, we use zero-knowledge proofs in several situations. On the one hand, we want voters to prove to the election authority that their cast ballot has certain properties (such as being a valid vote), without revealing the chosen voting option. On the other hand, we want the election authority to prove that they are correctly tallying the ballots maintaining some of the operations private.

The first zero-knowledge protocols [GMR85] were interactive, i.e., protocols between two entities (a Prover and a Verifier) in which each entity sends several messages, which depend on previously received messages, to the other entity. It isn't until [BFM88] when the first non-interactive zero-knowledge (NIZK) proofs were proposed, at the cost of assuming some prior shared information between prover and verifier.

In both of our electronic voting use cases, we want to use the non-interactive version of zero-knowledge proofs. In the first case, using non-interactive zero-knowledge proofs allows us to minimize the interaction between voters and election authorities and to be able to verify these proofs at any point during the election. In the second case, using non-interactive proofs means that the election authorities can produce a single proof for all stakeholders of the election, instead of having to interact with each stakeholder of the election individually. Therefore, we will mainly focus on NIZK proofs.

2.7.1 Definitions

Let us formalize what zero-knowledge proofs are. First, we define what sort of statements zero-knowledge proofs allow us to deal with. Then, we formalize the concepts of *not revealing any information other than the truthfulness of the statement* and *convincing the other entity that the statement is true*. The definitions that we use are mainly based on those given in [GOS06].

Consider a relation of triplets (gk, x, w). We call gk the setup, x the statement and w the witness. We can think of the witness as the evidence for the truth of the statement. The ternary relation, which is denoted by R, must be efficiently computable. In other words, it must be possible to check whether a tuple (gk, x, w) is in the relation in time polynomial in the length of the statement.

Now consider the language L_{gk} formed by those statements x such that there exists some witness w for which $(gk, x, w) \in R$, i.e., those statements for which there exists a witness for the truth of the statement. For a relation that ignores the setup gk, this is the standard definition of an NP-language. However, it will be useful to consider languages that might be parametrized by a setup gk which, for example, could be the description of a bilinear group.

Non-interactive proof systems

Before defining what a NIZK proof is, let us define the notion of non-interactive proof systems, which is primitive a simpler than NIZK proofs sharing its syntactical definition.

A non-interactive proof system for a relation R consists of four (possibly probabilistic) algorithms:

$\mathsf{Setup}(1^\lambda)$: on input a security parameter 1^λ, it outputs a setup gk.

$\mathsf{GenCRS}(gk)$: on input a setup gk it outputs a common reference string crs.

$\mathsf{Prove}(gk, \mathsf{crs}, x, w)$**:** on input a setup gk, a common reference string crs, a statement x and a witness w such that $(gk, x, w) \in R$ outputs a proof π.

$\mathsf{VerifyProof}(gk, \mathsf{crs}, x, \pi)$**:** on input a setup gk, a common reference string crs, a statement x and a proof π outputs 1 for success or 0 for failure.

Note that the common reference string (crs), known by both the prover and the verifier, is required in order to be able to build non-interactive zero-knowledge proofs for *interesting* relations R. Indeed, in [Ore87] it is shown that if we restrict to settings without common reference string it is impossible to build NIZK proofs for languages outside the complexity class BPP (the class of statements for which a witness can be efficiently found with a bounded error probability), unless one uses the Random Oracle Model.

The basic requirement of a non-interactive proof system is that the scheme works as intended if all participants are honest. This requirement is known as *completeness*.

Definition 2.7.1. *We say that a non-interactive proof system* (Setup, GenCRS, Prove, VerifyProof) *is complete for a relation R if for all setups gk generated by the* Setup *algorithm, all common reference strings* crs *generated by the* GenCRS *algorithm and, for all (x, w) such that $(gk, x, w) \in R$, it holds that* $\mathsf{VerifyProof}(gk, \mathsf{crs}, x, \mathsf{Prove}(gk, \mathsf{crs}, x, w)) = 1$.

Soundness and knowledge

The first security requirement of a non-interactive proof system is that it is sound: if the honest verifier accepts the proof then it must mean that the statement is true. No cheating prover should be able to convince the verifier if the statement is false. The security notion is formalized via the following experiment between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

1. First, the challenger generates a setup gk using algorithm Setup and a common reference string crs using the algorithm GenCRS and forwards (gk, crs) to the adversary.

2. Then, the adversary outputs a statement x and a proof π. The output of the experiment is a bit $\alpha = 1$ if $\mathsf{VerifyProof}(gk, \mathsf{crs}, x, \pi)$=1 and $x \notin L_{gk}$ and is 0 otherwise.

The value α represents the success of the adversary: when $\alpha = 1$ the adversary managed to output a false convincing statement. There are two flavors of soundness: perfect and computational. We speak about perfect soundness when the probability of α being 1 at the end of the game is exactly 0, i.e., the probability of the adversarial success is exactly 0. On the other hand, computational security is based on some computational assumption and a negligible probability of adversarial success (i.e., a negligible probability of α being 1 at the end of the game).

Definition 2.7.2. *A non-interactive proof system* (Setup, GenCRS, Prove, VerifyProof) *is* computationally sound *if, for all p.p.t. adversaries, the following advantage of the adversary in the game defined above is negligible:*

$$\mathbf{Adv}^{\textsf{soundness}}(\lambda) := \Pr[\alpha = 1]$$

We say that the non-interactive proof system is perfectly sound *if the advantage of any adversary (not restricted to probabilistic polynomial time) is exactly 0.*

Observe that this property is only stating that no cheating prover should convince an honest verifier if the statement is false. In particular, it does imply that the prover should know any witness at all when creating any convincing proof. There is another security notion which states that if the prover convinces the verifier then the prover must know a witness for the statement. This implies that the statement must be true, which makes this property stronger than soundness.

Showing that a prover knows a witness might be hard. For this reason, the security notion asks that proof *contains* the witness (in a hidden way). The security notion works as follows. First, it requires the existence of an alternative common reference string generator ExtGenCRS which outputs an *extraction key xk* in addition to the common reference string crs. This extraction key is used to extract witnesses from proofs by means of an extraction algorithm Extract. The distribution of common reference strings crs produced by the alternative generator ExtGenCRS must be identically distributed to the distribution of common reference strings crs output by GenCRS.

In some cases, being able to extract a function $\mathcal{F}$ of the witness is enough for the NIZK proof to be used as a building block of a cryptographic protocol. In this case, the property is called $\mathcal{F}$-knowledge. Knowledge, the property of being able to extract the full witness, is a particular case of $\mathcal{F}$-knowledge where $\mathcal{F}$ is set to be the identity function. The security notion is formalized via the following experiment between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, parametrized by algorithms ExtGenCRS and Extract:

1. First, the challenger generates a setup gk with the algorithm Setup and then runs ExtGenCRS to obtain a common reference string crs and an extraction key xk. It keeps the extraction key xk and forwards the setup gk and the common reference string crs to the adversary.

2. Then, the adversary $\mathcal{A}$ outputs a statement x and a proof π.

3. Finally, the challenger $\mathcal{C}$ obtains an element $\tilde{f} = \mathsf{Extract}(gk, \mathsf{crs}, x, xk, \pi)$. The output of the experiment is a bit $\beta = 1$ if $\mathsf{VerifyProof}(gk, \mathsf{crs}, x, \pi)$=1 and there exists no w^* such that $\mathcal{F}(w^*) = \tilde{f}$ and $(gk, x, w^*) \in R$ and is 0 otherwise.

Similar to the soundness property, $\mathcal{F}$-knowledge can be perfect or computational.

Definition 2.7.3. *A non-interactive proof system* (Setup, GenCRS, Prove, VerifyProof) *is* computationally $\mathcal{F}$-extractable *if there exist algorithms* ExtGenCRS, Extract *such that the common reference strings* crs *generated by* GenCRS *and the common reference strings* crs *generated by* $\mathsf{ExtGenCRS}$ *are identically distributed and, for all p.p.t. adversaries, the following advantage of the adversary in the game defined above is negligible:*

$$\mathbf{Adv}^{\mathcal{F}\text{-}ext}_{\mathsf{ExtGenCRS},\mathsf{Extract}}(\lambda) := \Pr[\beta = 1]$$

We say that the non-interactive proof system is perfectly $\mathcal{F}$-extractable *if the distribution of the* crs *generated by* $\mathsf{ExtGenCRS}$ *and the distribution of the* crs *generated by* GenCRS *are identical and the advantage of the adversary defined as above is 0.*

When a non-interactive proof system is $\mathcal{F}$-extractable (resp. extractable), we say that the proof system is a non-interactive proof of $\mathcal{F}$-knowledge (resp. of knowledge).

Witness-Indistinguishability and Zero-Knowledge

The Witness-Indistinguishability property states that given a proof it should be impossible to distinguish which witness was used to create the proof. In other words, proofs created with a witness w_1 should be indistinguishable from proofs created by using a different witness w_2.

The security notion requires the existence of another crs generation algorithm, $\mathsf{WIGenCRS}$, which output distribution must be computationally indistinguishable from the distribution of crs generated by the GenCRS algorithm.

Formally, the security notion is formalized via the following experiment between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, parametrized by an algorithm $\mathsf{WIGenCRS}$:

1. First, the challenger generates a setup gk via the algorithm Setup and a common reference string crs using the algorithm $\mathsf{WIGenCRS}$. It forwards (gk, crs) to the adversary.
2. Then, the adversary outputs a statement x and two valid witnesses w_0, w_1.
3. The challenger draws a random bit $b \leftarrow \{0, 1\}$ and computes the proof as $\pi = \mathsf{Prove}(gk, \mathsf{crs}, x, w_b)$. The challenger submits the proof π to the adversary.
4. Finally, the adversary submits a bit $\tilde{b}$.

Just like soundness and $\mathcal{F}$-extractability, witness-indistinguishability comes in two variants: perfect witness-indistinguishability and computational witness-indistinguishability.

Definition 2.7.4. *A non-interactive proof system* $(\mathsf{Setup}, \mathsf{GenCRS}, \mathsf{Prove}, \mathsf{VerifyProof})$ *is* computationally witness-indistinguishable *if there exists an algorithm* $\mathsf{WIGenCRS}$ *such that the distribution of crs output by* GenCRS *algorithm is computationally indistinguishable from the distribution of crs output by* $\mathsf{WIGenCRS}$ *and, for all p.p.t. adversaries, the following advantage of the adversary in the game defined above is 0:*

$$\mathbf{Adv}^{WI}_{\mathsf{WIGenCRS}}(\lambda) := |\Pr[\tilde{b} = b] - 1/2|$$

We say that a non-interactive proof system is perfectly witness-indistinguishable *if the distribution of the crs generated by* GenCRS *and the distribution of the crs generated by* $\mathsf{WIGenCRS}$ *are identical and the advantage of any adversary defined as above is 0.*

When a non-interactive proof system has the Witness-Indistinguishability property, we say that it is a Non-Interactive Witness-Indistinguishable (NIWI) proof.

The zero-knowledge property states the only information given by the proof should be the statement's truth. For instance, witness-indistinguishability might not prevent a proof from revealing the set of all possible witnesses, while that would be not possible with a zero-knowledge proof (the zero-knowledge property implies that finding any witness given a zero-knowledge proof is as hard as finding such witness without knowledge of the proof).

Formalizing the concept that the only information given by the proof is the truth of the statement is complex. This is why zero-knowledge is formalized in a different way. For the NIZK proof to be zero-knowledge there must exist (another) alternate crs generation algorithm, $\mathsf{SimGenCRS}$ which, in addition to the common reference string, outputs a trapdoor key tk. This trapdoor key can then be used to *simulate* proofs without the knowledge of the witness via a proof simulation algorithm $\mathsf{SimProve}$. The distribution of the crs generated by the $\mathsf{SimGenCRS}$ algorithm must be computationally indistinguishable from the distribution of crs generated by the GenCRS algorithm. Furthermore, the distribution of simulated proofs generated by the $\mathsf{SimProve}$ algorithm should be computationally indistinguishable from the distribution of real proofs generated by the Prove algorithm.

The reasoning behind this definition is the following. If the simulated proof looks exactly like an honestly generated proof and the simulated proof was created without any witness it must mean that honestly generated proofs do not give information about the witness. Note that the trapdoor key tk does not help in finding witnesses, it only allows the prover to simulate proofs. The indistinguishability requirements imply that no adversary can tell apart real from simulated proofs.

The security notion is formalized via the following experiment between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, parametrized by algorithms $\mathsf{SimGenCRS}, \mathsf{SimProve}$:

1. First, the challenger generates a setup gk using the algorithm Setup and then runs the algorithm $\mathsf{SimGenCRS}$ to generate a common reference string crs and a trapdoor key tk. It keeps the trapdoor key tk and forwards the setup gk and the common reference string crs to the adversary.

2. Then, the adversary outputs a statement x and a witness w.

3. The challenger draws a random bit $b \leftarrow \{0, 1\}$. If $b = 0$ then the challenger computes the proof using the *honest* proof generation algorithm: $\pi = \mathsf{Prove}(gk, \mathsf{crs}, x, w)$. If $b = 1$ then the challenger simulates the proof: $\pi = \mathsf{SimProve}(gk, \mathsf{crs}, tk, x)$. The challenger submits the proof π to the adversary.

4. Finally, the adversary submits a bit $\tilde{b}$.

Definition 2.7.5. *A non-interactive proof system* $(\mathsf{Setup}, \mathsf{GenCRS}, \mathsf{Prove}, \mathsf{VerifyProof})$ *is* computationally zero-knowledge *if there exist algorithms* SimGenCRS, SimProve *such that the distribution of common reference strings output by* GenCRS *algorithm is computationally indistinguishable from the distribution of common reference strings output by* SimGenCRS *and, for all p.p.t. adversaries, their advantage in the game above defined as follows is 0:*

$$\mathbf{Adv}^{\mathsf{ZK}}_{\mathsf{SimGenCRS},\mathsf{SimProve}}(\lambda) := |\Pr[\tilde{b} = b] - 1/2|$$

We say that a non-interactive proof system is perfectly zero-knowledge *if the distribution of the crs generated by* GenCRS *and the distribution of the crs generated by* SimGenCRS *are identical and the advantage of any adversary defined as above is 0.*

A straightforward argument shows that the zero-knowledge property is stronger than the witness-indistinguishability property. Indeed, if a non-interactive proof system is zero-knowledge then by defining the algorithm WIGenCRS as the algorithm SimGenCRS without outputting the trapdoor key tk we get a witness-indistinguishable non-interactive proof.

A non-interactive proof system with the zero-knowledge property is also called a Non-Interactive Zero-Knowledge (NIZK) proof. A NIZK proof with extractability is usually called a Non-Interactive Zero-Knowledge Proof of Knowledge (NIZK-PoK).

To be accurate, the notions of witness-indistinguishability and zero-knowledge that we have defined are called composable witness-indistinguishability and composable zero-knowledge [Gro06], which are stronger notions than the classical notions of witness-indistinguishability and zero-knowledge.

2.7.2 Non-interactive zero-knowledge from sigma protocols

One popular way of constructing Non-Interactive Zero-Knowledge protocols consists in modifying (interactive) Zero-Knowledge protocols to remove their interactivity. One of such methods is the so-called Fiat-Shamir heuristic, which transforms Sigma protocols into Non-Interactive Zero-Knowledge Proofs of Knowledge.

Sigma protocols and the Fiat-Shamir heuristic

Sigma (Σ) protocols are 3-move interactive protocols between a prover P and a verifier V, where P wants to prove knowledge of a witness w for a statement x. Formally, let R be a relation of tuples $(gk, x, w) \in \mathcal{R}$, where gk is a setup known to all parties, x will be a statement common to the prover P and to the verifier V and w will be a witness only known to the prover P. Σ-protocols have the following form:

1. P sends a message a.

2. V sends a random t-bit string c.

3. P sends a reply z, and V decides to accept or reject based on the data he has seen, i.e., x, a, c, z.

Definition 2.7.6. *A protocol $\mathcal{P}$ is said to be a Σ-protocol for a relation R if:*

- *$\mathcal{P}$ is of the above 3-move form, and we have completeness: if P and V follow the protocol honestly and their inputs are such that $(gk, x, w) \in R$ then V always accepts.*
- *$\mathcal{P}$ has the special soundness property: from any x and any pair of accepting conversations (a, c, z), (a, c', z') where $c \neq c'$, one can efficiently compute w such that $(gk, x, w) \in R$.*
- *$\mathcal{P}$ has the special honest-verifier zero-knowledge property: there exists a simulator M which on input x and a random c outputs an accepting conversation of the form (a, c, z) with the same probability distribution as the conversations between honest P, V on input x.*

An example of such protocol is the Scnorr protocol [Sch91], which is used to prove knowledge of discrete logarithms in a group $(\bar{\mathbb{G}}, p, \bar{g})$. Let $\bar{h} \in \bar{\mathbb{G}}$ be a statement, known to both the prover and the verifier, and let w be a witness such that $\bar{h} = w\bar{g}$. The protocol for proving knowledge of such w works as follows:

1. P chooses $r \leftarrow \mathbb{Z}_p$ and sends $\bar{a} = r\bar{g}$ to V.

2. V chooses a challenge $c \leftarrow \mathbb{Z}_{2^t}$ and sends it to P, where t is such that $2^t < p$.

3. P sends $z = r + cw \mod p$ to V, who checks that $z\bar{g} = \bar{a} + c\bar{h} \in \bar{\mathbb{G}}$, and accepts iff this is the case.

It is straightforward to check that the properties required for the protocol to be a Σ-protocol are satisfied. Indeed, the protocol is of the mentioned 3-move form. Completeness and special soundness can be verified by using basic algebra. Finally, the simulator M on input $(\bar{h}, c)$ gets $z \leftarrow \mathbb{Z}_p$ and computes $\bar{a} = z\bar{g} - c\bar{h} \in \bar{\mathbb{G}}$, creating a transcript $(\bar{a}, c, z)$ identically distributed to those resulting from executing the protocol.

Σ-protocol have many applications, some of them described in [Dam]. Among others, we can use Σ-protocols to construct efficient NIZK-PoK in the Random Oracle Model, as described in [FS86].

Let $\mathcal{O}$ be an oracle $\mathcal{O} : \bar{\mathbb{G}} \to \mathbb{Z}_{2^t}$. To remove interactivity from the Σ-protocol, the prover will need to replace the challenge c with the response from the random oracle on input the statement x and the message a, and compute the last message z as it would do in the interactive protocol. As the random oracle is known to both the prover and the verifier, the prover only needs to send (a, z) to the verifier, who can compute $c = \mathcal{O}(x, a)$ and check the verification equation. Formally, we define the non-interactive proof as follows:

$\mathsf{Setup}(1^\lambda)$**:** on input a security parameter 1^λ, it outputs any setup gk required by the Σ-protocol, and a random oracle $\mathcal{O} : \{0,1\}^* \to \{0,1\}^t$.

$\mathsf{Prove}(gk, x, w)$**:** on input a setup gk, a statement x and a witness w does the following steps:

1. Computes the value a as in the Σ-protocol
2. Queries the random oracle $\mathcal{O}$ on the statement x and the message a, obtaining $c = \mathcal{O}(x, a)$
3. Computes the value z as if it had received c from the verifier V.

The output of this algorithm is the proof $\pi = (a, z)$.

$\mathsf{VerifyProof}(gk, x, \pi)$**:** on input a setup gk, a statement x and a proof $\pi = (a, z)$ queries the oracle $c = \mathcal{O}(x, a)$ and outputs 1 if the tuple (x, a, c, z) satisfies the Σ-protocol verification relation, otherwise outputs 0.

Note that the absence of the common reference string crs does not contradict the results in [Ore87], since those results do not contemplate the existence of a random oracle.

In [BPW12a], it is shown that this construction, called the strong Fiat-Shamir construction, results in a Non-Interactive Zero-Knowledge Proof of Knowledge, i.e., it satisfies completeness, zero-knowledge and extractability. The results show that the construction results in a simulation-sound extractable protocol, a stronger notion than extractability.

2.8 Groth-Sahai proofs

Early NIZK proof constructions with security not relying on the ROM [BFM88, FLS99, Dam, KP98] were very inefficient. This changed when Groth, Ostrovsky and Sahai [GOS12] introduced pairing-based techniques for constructing NIZK proofs. In a series of works [BW06, Gro06, BW07, GS12], pairing-friendly NIZK proofs were developed. This line of research culminated in Groth and Sahai's paper [GS12], that gives a framework for building efficient and practical NIZK proofs that are now widely used in pairing-based cryptography.

Known instantiations of the Groth-Sahai framework are based on bilinear groups $(p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathcal{PG}$ and are used to build proofs for the language of satisfiability of set of the following equations on variables $\hat{x}_i \in \hat{\mathbb{G}}, \check{y}_j \in \check{\mathbb{H}}, x_i, y_j \in \mathbb{Z}_p$:

Pairing-product equation: Public constants $\hat{a}_j \in \hat{\mathbb{G}}, \check{b}_i \in \check{\mathbb{H}}, \gamma_{ij} \in \mathbb{Z}_p, t_{\mathbb{T}} \in \mathbb{T}$.

$$\sum_i e(\hat{x}_i, \check{b}_i) + \sum_j e(\hat{a}_j, \check{y}_j) + \sum_i \sum_j e(\gamma_{ij} \hat{x}_i, \check{y}_j) = t_{\mathbb{T}}.$$

Multi-scalar multiplication equation in $\hat{\mathbb{G}}$: Public constants $\hat{a}_j \in \hat{\mathbb{G}}$, $b_i \in \mathbb{Z}_p$, $\gamma_{ij} \in \mathbb{Z}_p, \hat{t} \in \hat{\mathbb{G}}$.

$$\sum_i \hat{x}_i b_i + \sum_j \hat{a}_j y_j + \sum_i \sum_j \gamma_{ij} \hat{x}_i y_j = \hat{t}.$$

Multi-scalar multiplication equation in $\check{\mathbb{H}}$: Public constants $a_j \in \mathbb{Z}_p, \check{b}_i \in \check{\mathbb{H}}, \gamma_{ij} \in \mathbb{Z}_p, \check{t} \in \check{\mathbb{H}}$.

$$\sum_i x_i \check{b}_i + \sum_j a_j \check{y}_j + \sum_i \sum_j \gamma_{ij} x_i \check{y}_j = \check{t}.$$

Quadratic equation in $\mathbb{Z}_p$: Public constants $a_j \in \mathbb{Z}_p, b_i \in \mathbb{Z}_p, \gamma_{ij} \in \mathbb{Z}_p, t \in \mathbb{Z}_p$.

$$\sum_i x_i b_i + \sum_j a_j y_j + \sum_i \sum_j \gamma_{ij} x_i y_j = t.$$

These four types of equations express in a direct way statements arising in pairing-based cryptography. For this reason Groth-Sahai proofs are used in numerous pairing-based protocols including group signatures [Gro07], anonymous credentials [BCKL08, BCC$^+$09], e-cash [FPV09], etc.

Although we presented the equations above for asymmetric bilinear groups, Groth-Sahai proofs admit bilinear groups of type-I, type-II (due to an extension given in [GSW10]) and type-III. Depending on which type of bilinear groups is used, concrete Groth-Sahai proofs will be instantiated from the Groth-Sahai proof framework differently and their security might be based on different assumptions.The setting which yields the most efficient proofs are type-III bilinear groups.

Before explaining how Groth-Sahai proofs are constructed, let us briefly introduce the notion of a commitment scheme. A commitment scheme is a cryptographic primitive which allows to commit to a chosen value, by producing a commitment which hides the value to other entities. Then, at a later stage, the commitment can be opened to reveal the hidden value. Commitments are created by using a *commitment key* and have two properties: the binding property, which states that it should be unfeasible to open the commitment to a value different than the original one, and the hiding property, which states that the commitment should not reveal any information about the committed value. As with zero-knowledge proofs, these properties come in two flavors: either a commitment scheme is computationally binding and perfectly hiding or it is perfectly binding and computationally hiding.

2.8.1 Overview

The main ideas behind the Groth-Sahai framework are the following.

To provide soundness and witness-indistinguishability the common reference string might be generated in two possible settings: the soundness setting, from which soundness and extractability properties will follow, and the hiding setting, from which the witness-indistinguishability and the zero-knowledge properties will follow. These two settings are computationally indistinguishable, therefore achieving all properties (some of them are in a computational fashion).

The common reference string specifies some commitment keys to commit to variables, both group elements and scalars. When the common reference string is set in a soundness setting, these commitments will be perfectly binding, whereas when the setting is the hiding one the commitments will be perfectly hiding.

These commitments will satisfy some related set of equations over bilinear groups. However, to be able to verify these equations, some elements related to the randomness used in the commitments will be needed in the proof. These elements are randomized so that the related set of equations still hold but no further information is revealed.

2.8.2 The Groth-Sahai proof framework

The Groth-Sahai framework abstracts the equations given in Section 2.8 as equations in $\mathcal{R}$-modules $\mathfrak{A}_1, \mathfrak{A}_2$ and $\mathfrak{A}_T$ with a bilinear map $f : \mathfrak{A}_1 \times \mathfrak{A}_2 \to \mathfrak{A}_T$.

In the case of pairing-product equations, the ring $\mathcal{R}$ is $\mathbb{Z}_p$, the modules $\mathfrak{A}_1, \mathfrak{A}_2$ and $\mathfrak{A}_T$ are $\hat{\mathbb{G}}, \check{\mathbb{H}}$ and $\mathbb{T}$ respectively, and the bilinear map f is the bilinear map e. Multi-linear equations in $\hat{\mathbb{G}}$ (resp. in $\check{\mathbb{H}}$) are obtained by having the ring $\mathcal{R}$ be $\mathbb{Z}_p$, the modules $\mathfrak{A}_1, \mathfrak{A}_T$ (resp. $\mathfrak{A}_2, \mathfrak{A}_T$) be $\hat{\mathbb{G}}$ (resp. $\check{\mathbb{H}}$), the module $\mathfrak{A}_2$ (resp. $\mathfrak{A}_1$) be $\mathbb{Z}_p$ and the bilinear map f be the product of a scalar in $\mathbb{Z}_p$ by a group element in $\hat{\mathbb{G}}$ (resp. $\check{\mathbb{H}}$). Finally, scalar quadratic equations in $\mathbb{Z}_p$ are obtained by having the ring $\mathcal{R}$ be $\mathbb{Z}_p$, the modules $\mathfrak{A}_1, \mathfrak{A}_2$ and $\mathfrak{A}_T$ be $\mathbb{Z}_p$ and the map f be the product of scalars in $\mathbb{Z}_p$.

To simplify notation, let us for $\mathbf{x} \in \mathfrak{A}_1^n$, $\mathbf{y} \in \mathfrak{A}_2^n$ define

$$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{n} f(x_i, y_i)$$

The equations given in Section 2.8 can now be written as

$$\mathbf{a} \cdot \mathbf{y} + \mathbf{x} \cdot \mathbf{b} + \mathbf{x} \cdot \Gamma \mathbf{y} = t,$$

where $\mathbf{a} \in \mathfrak{A}_1^n, \mathbf{b} \in \mathfrak{A}_2^m, \Gamma \in \mathcal{R}^{m \times n}, t \in \mathfrak{A}_T$. Note that a bilinear equation is completely specified by $(\mathbf{a}, \mathbf{b}, \Gamma, t)$.

The Groth-Sahai proof framework for satisfiability of quadratic equations in $(\mathfrak{A}_1, \mathfrak{A}_2, \mathfrak{A}_T, f)$ specifies the following requirements and constructions for the algorithms of a NIZK proof:

$\mathsf{Setup}(1^\lambda)$**:** on input the security parameter 1^λ, it must output a setup gk, which consists of a finite commutative ring $\mathcal{R}$, the $\mathcal{R}$-modules $\mathfrak{A}_1, \mathfrak{A}_2, \mathfrak{A}_T$ and the bilinear map $f : \mathfrak{A}_1 \times \mathfrak{A}_2 \to \mathfrak{A}_T$.

$\mathsf{GenCRS}(gk)$**:** on input the setup gk outputs the common reference string crs. The common reference string must contain some $\mathcal{R}$-modules $\mathfrak{B}_1, \mathfrak{B}_2, \mathfrak{B}_T$, a bilinear map $F : \mathfrak{B}_1 \times \mathfrak{B}_2 \to \mathfrak{B}_T$, maps $\iota_i : \mathfrak{A}_i \to \mathfrak{B}_i$ and $\mathfrak{p}_i : \mathfrak{B}_i \to \mathfrak{A}_i$ for $i \in \{1, 2, T\}$, commitment keys $\mathbf{u} \in \mathfrak{B}_1^{1 \times \hat{m}}$,$\mathbf{v} \in \mathfrak{B}_2^{1 \times \hat{n}}$ and matrices $\{H_i\}_{i=1}^{\eta} \in \mathcal{R}^{\hat{m} \times \hat{n}}$. These elements must satisfy the following properties:

1. The maps $\iota_1, \mathfrak{p}_1, \iota_2, \mathfrak{p}_2, \iota_T, \mathfrak{p}_T, F$ and f must commute as follows:

$$\forall x \in \mathfrak{A}_1, \forall y \in \mathfrak{A}_2 : F(\iota_1(x), \iota_2(y)) = \iota_T(f(x, y)),$$
$$\forall x \in \mathfrak{B}_1, \forall y \in \mathfrak{B}_2 : f(\mathfrak{p}_1(x), \mathfrak{p}_2(y)) = \mathfrak{p}_T(F(x, y)).$$

2. The maps ι_1,ι_2, ι_T and F must be efficiently computable.
3. $\mathfrak{p}_1(\mathbf{u}) = \mathbf{0}$, $\mathfrak{p}_2(\mathbf{v}) = \mathbf{0}$ and the maps $\mathfrak{p}_1 \circ \iota_1$, $\mathfrak{p}_2 \circ \iota_2$ and $\mathfrak{p}_T \circ \iota_T$ are the identity map in their respective domains[1].

We have not imposed yet any condition on the matrices $\{H_i\}_{i=1}^{\eta}$. These conditions are imposed on the WIGenCRS *algorithm, which we will later give. Indistinguishability between* GenCRS *and* WIGenCRS *will indirectly impose conditions on $\{H_i\}_{i=1}^{\eta}$.*

$\mathsf{Prove}(gk, \mathsf{crs}, x, w)$**:** Takes as input the setup gk, the common reference string crs, a statement, which is a list of quadratic equations $\{\mathbf{a}_i, \mathbf{b}_i, \Gamma_i, t_i\}_{i=1}^{N}$, and a satisfying witness $\mathbf{x} \in \mathfrak{A}_1^m, \mathbf{y} \in \mathfrak{A}_2^n$. It computes the proof as follows:

1. Picks uniformly at random $R \leftarrow \mathcal{R}^{\hat{m}\times m}$ and $S \leftarrow \mathcal{R}^{\hat{n}\times n}$ and computes commitments to variables $\mathbf{x}, \mathbf{y}$ as $\mathbf{c} = \iota_1(\mathbf{x}^\top) + \mathbf{u}R \in \mathfrak{B}_1^{1\times m}$ and $\mathbf{d} = \iota_2(\mathbf{y}^\top) + \mathbf{v}S \in \mathfrak{B}_2^{1\times n}$, where ι_1, ι_2 operate on $\mathbf{x}^\top, \mathbf{y}^\top$ element-wise.
2. For each equation $(\mathbf{a}_i, \mathbf{b}_i, \Gamma_i, t_i)$ computes its proof by picking $T_i \leftarrow \mathcal{R}^{\hat{m}\times\hat{n}}$, $r_{i1}, \ldots, r_{i\eta} \leftarrow \mathcal{R}$ and computing:

$$\begin{aligned}\boldsymbol{\pi}_i &= \left(\iota_2(\mathbf{b}_i^\top) + \iota_2(\mathbf{y}^\top)\Gamma_i^\top + \mathbf{v}S\Gamma_i^\top\right) R^\top - \mathbf{v}T_i^\top + \sum_{j=1}^{\eta} r_{ij}\mathbf{v}H_i^\top \in \mathfrak{B}_2^{1\times\hat{m}} \\ \boldsymbol{\theta}_i &= \left(\iota_1(\mathbf{a}_i^\top) + \iota_1(\mathbf{x}^\top)\Gamma_i\right) S^\top + \mathbf{u}T_i \in \mathfrak{B}_1^{1\times\hat{n}}\end{aligned}$$

3. Outputs the proof, defined as $(\mathbf{c}, \mathbf{d}, \{(\boldsymbol{\pi}_i, \boldsymbol{\theta}_i\}_{i=1}^{N})$.

$\mathsf{VerifyProof}(gk, \mathsf{crs}, x, \pi)$**:** takes as input the setup gk, the common reference string crs, a set of equations $\{\mathbf{a}_i, \mathbf{b}_i, \Gamma_i,\, t_i\}_{i=1}^{N}$ and a proof $(\mathbf{c}, \mathbf{d}, \{(\boldsymbol{\pi}_i, \boldsymbol{\theta}_i\}_{i=1}^{N})$. For notational convenience let us define for $\mathbf{x} \in \mathfrak{B}_1^{1\times n}, \mathbf{y} \in \mathfrak{B}_2^{1\times n}$: $\mathbf{x} \bullet \mathbf{y} = \sum_{i=1}^{n} F(x_i, y_i)$. For each equation, the VerifyProof algorithm checks:

$$\iota_1(\mathbf{a}_i^\top) \bullet \mathbf{d} + \mathbf{c} \bullet \iota_2(\mathbf{b}_i^\top) + \mathbf{c} \bullet \mathbf{d}\Gamma_i^\top = \iota_T(t_i) + \mathbf{u} \bullet \boldsymbol{\pi}_i + \boldsymbol{\theta}_i \bullet \mathbf{v},$$

and outputs 1 if all checks pass, otherwise it outputs 0.

Linear equations A special case occurs when $\mathbf{a} = \mathbf{0}$ and $\Gamma = 0$. In this case, the statement equation is simply

$$\mathbf{x} \cdot \mathbf{b} = t.$$

The prover can generate shorter proofs by choosing $T = 0$, which gives $\boldsymbol{\pi}_i = \iota_2(\mathbf{b}^\top)R^\top + \sum_{j=1}^{\eta} r_{ij}\mathbf{v}H_i^\top$ and $\boldsymbol{\theta} = \mathbf{0}$. The verification equation remains the same, but there is no need to compute $\boldsymbol{\theta} \bullet \mathbf{v} = \mathbf{0}$.

[1] This requirement is relaxed in [GS12], where it is asked that the compositions of those maps are non-trivial. If this composition is not the identity map, then soundness can not be achieved with the Groth-Sahai framework.

The symmetric case Another special case is when $\mathfrak{B} := \mathfrak{B}_1 = \mathfrak{B}_2$, $\hat{m} \geq \hat{n}$ with $u_1 = v_1, \cdots, u_{\hat{m}} = v_{\hat{m}}$, and for all $x, y \in \mathfrak{B}$ we have $F(x,y) = F(y,x)$. This is called the symmetric case. In the symmetric case, the proof can be simplified by padding $\boldsymbol{\theta}$ with zeros in the end to extend its length to $\hat{m}$, call this vector $\boldsymbol{\theta}'$, and reveal the proof $\phi = \boldsymbol{\pi} + \boldsymbol{\theta}'$. The verification equation is then

$$\iota_1(\mathbf{a}^\top) \bullet \mathbf{d} + \mathbf{c} \bullet \iota_2(\mathbf{b}^\top) + \mathbf{c} \bullet \mathbf{d}\Gamma^\top = \iota_T(t) + \mathbf{u} \bullet \phi.$$

The Groth-Sahai proof framework also specifies that an algorithm $\mathsf{WIGenCRS}$ must exist, with the following properties:

$\mathsf{WIGenCRS}(gk)$: on input the setup gk outputs the common reference string crs. The common reference string must contain some $\mathcal{R}$-modules $\mathfrak{B}_1, \mathfrak{B}_2, \mathfrak{B}_T$, a bilinear map $F : \mathfrak{B}_1 \times \mathfrak{B}_2 \to \mathfrak{B}_T$, maps $\iota_i : \mathfrak{A}_i \to \mathfrak{B}_i$ and $\mathfrak{p}_i : \mathfrak{B}_i \to \mathfrak{A}_i$ for $i \in \{1, 2, T\}$, commitment keys $\mathbf{u} \in \mathfrak{B}_1^{1\times\hat{m}}$,$\mathbf{v} \in \mathfrak{B}_2^{1\times\hat{n}}$ and matrices $\{H_i\}_{i=1}^{\eta} \in \mathcal{R}^{\hat{m}\times\hat{n}}$. These elements must satisfy the following properties:

1. The maps $\iota_1, \mathfrak{p}_1, \iota_2, \mathfrak{p}_2 \iota_T, \mathfrak{p}_T, F$ and f must commute as follows:

$$\forall x \in \mathfrak{A}_1, \forall y \in \mathfrak{A}_2 : F(\iota_1(x), \iota_2(y)) = \iota_T(f(x,y)),$$
$$\forall x \in \mathfrak{B}_1, \forall y \in \mathfrak{B}_2 : f(\mathfrak{p}_1(x), \mathfrak{p}_2(y)) = \mathfrak{p}_T(F(x,y)).$$

2. The maps ι_1,ι_2, ι_T and F must be efficiently computable.
3. $\iota_1(\mathfrak{A}_1) \subseteq \langle u_1, \ldots, u_{\hat{m}} \rangle$ and $\iota_1(\mathfrak{A}_2) \subseteq \langle v_1, \ldots, v_{\hat{n}} \rangle$.
4. $H_1, \ldots, H_\eta$ generate the $\mathcal{R}$-module of all matrices $H \in \mathcal{R}^{\hat{m}\times\hat{n}}$ such that $\mathbf{u}H \bullet \mathbf{v} = 0$.

In addition to the restrictions mentioned above, the Groth-Sahai proof framework specifies that the outputs of GenCRS and $\mathsf{WIGenCRS}$ must be computationally indistinguishable.

In [GS12], they state and prove the following theorem:

Theorem 2.8.1. *The proof system* $(\mathsf{Setup}, \mathsf{GenCRS}, \mathsf{Prove}, \mathsf{VerifyProof})$ *satisfying the restrictions mentioned above is a NIWI proof for satisfiability of a set of quadratic equations with perfect completeness, perfect soundness and composable witness-indistinguishability if there exists an algorithm* $\mathsf{WIGenCRS}$ *such that* Setup, GenCRS, $\mathsf{WIGenCRS}$, Prove, $\mathsf{VerifyProof}$ *are built as specified above and the output generated by* GenCRS *is computationally indistinguishable from the output generated by* $\mathsf{WIGenCRS}$.

2.8.3 NIWI proofs for bilinear groups

So far, we have defined how to create NIWI proofs for satisfiability of systems of equations in generic $\mathcal{R}$-modules with a bilinear map $(\mathfrak{A}_1, \mathfrak{A}_2, \mathfrak{A}_T, f)$. However, we are interested in how to create NIWI proofs for a set of quadratic equations over bilinear groups, where we can have a combination of:

Pairing product equations: $\mathfrak{A}_1 = \hat{\mathbb{G}}, \mathfrak{A}_2 = \check{\mathbb{H}}, \mathfrak{A}_T = \mathbb{T}, f(\hat{x}, \check{y}) = e(\hat{x}, \check{y})$.

Multiscalar multiplication in $\hat{\mathbb{G}}$: $\mathfrak{A}_1 = \hat{\mathfrak{A}}_T = \hat{\mathbb{G}}, \mathfrak{A}'_2 = \mathbb{Z}_p, \hat{f}(\hat{x}, y) = y\hat{x}$

Multiscalar multiplication in $\check{\mathbb{H}}$: $\mathfrak{A}_2 = \check{\mathfrak{A}}_T = \check{\mathbb{H}}, \mathfrak{A}'_1 = \mathbb{Z}_p, \check{f}(x, \check{y}) = x\check{y}$

Quadratic equations in $\mathbb{Z}_p$: $\mathfrak{A}'_1 = \mathfrak{A}'_2 = \mathfrak{A}'_T = \mathbb{Z}_p, f'(x, y) = xy \mod p.$

The first approach would be to have four NIWI protocols, one for each equation type. However, to guarantee that the same values for the variables are consistently used across all equations, the same commitments (and commitment schemes) must be used across all equations.

Therefore, we will have at most four different modules $\mathfrak{B}_1, \mathfrak{B}'_1, \mathfrak{B}_2, \mathfrak{B}'_2$ to commit to variables in $\mathfrak{A}_1, \mathfrak{A}'_1, \mathfrak{A}_2, \mathfrak{A}'_2$ respectively. Similarly, there will be at most four inclusion maps $\iota_1, \iota'_1, \iota_2, \iota'_2$, four projection maps $\mathfrak{p}_1, \mathfrak{p}'_1, \mathfrak{p}_2, \mathfrak{p}'_2$ and four sets of commitment keys $\mathbf{u}, \mathbf{u}', \mathbf{v}, \mathbf{v}'$.

In principle, the Groth-Sahai framework allows having a different $\mathfrak{B}_T$ module for each equation type. Therefore, we could have a module $\mathfrak{B}_T$ for pairing product equations, with the corresponding maps F, ι_T, $\mathfrak{p}_T$, a module $\hat{\mathfrak{B}}_T$ (resp. $\check{\mathfrak{B}}_T$) for multi-scalar multiplications in $\hat{\mathbb{G}}$ (resp. in $\check{\mathbb{H}}$) and corresponding maps $\hat{F}, \hat{\iota}_T, \hat{\mathfrak{p}}_T$ (resp. $\check{F}, \check{\iota}_T, \check{\mathfrak{p}}_T$) and a module $\mathfrak{B}'_T$ for quadratic equations in $\mathbb{Z}_p$ and corresponding maps $F', \iota'_T, \mathfrak{p}'_T$.

However, in all currently known Groth-Sahai proof instantiations, the modules $\tilde{\mathfrak{B}}_1, \tilde{\mathfrak{B}}_2$ and $\tilde{\mathfrak{B}}_T$ are the same for all equation types. In addition, in asymmetric bilinear groups only one map $F : \mathfrak{B}_1 \times \mathfrak{B}_2 \to \mathfrak{B}_T$ is used for all four equation types. In symmetric bilinear groups two maps $F, F_{sym} : \mathfrak{B}_1 \times \mathfrak{B}_2 \to \mathfrak{B}_T$ are usually defined for efficiency reasons as we will show later in this section, when we explain one of the instantiations of Groth-Sahai proofs given in [GS12].

In summary, in a typical Groth-Sahai proof instantiation, the GenCRS and WIGenCRS algorithms will need to output:

- $\mathbb{Z}_p$-modules $\mathfrak{B}_1$, $\mathfrak{B}_2$ where variables will be committed to.
- A $\mathbb{Z}_p$-module $\mathfrak{B}_T$, a bilinear map F and, possibly, its symmetric variant F_{sym}.
- Inclusion maps $\iota_1, \iota'_1, \iota_2, \iota'_2$, projection maps $\mathfrak{p}_1, \mathfrak{p}'_1, \mathfrak{p}_2, \mathfrak{p}'_2$ and commitment keys $\mathbf{u}, \mathbf{u}', \mathbf{v}, \mathbf{v}'$, to construct commitments to group elements and scalars.
- For each equation type:
  - The corresponding inclusion and projection maps $\tilde{\iota}_T$, $\tilde{\mathfrak{p}}_T$ and their symmetric versions if needed.
  - The set of matrices $\{\tilde{H}_i\}$ such that, when the commitment keys $\tilde{\mathbf{u}}$ (either $\mathbf{u}$ or $\mathbf{u}'$, depending on the equation type) and $\tilde{\mathbf{v}}$ (either $\mathbf{u}$ or $\mathbf{u}'$) are generated by WIGenCRS generate all matrices H such that $\tilde{\mathbf{u}} \bullet H\tilde{\mathbf{v}} = 0$.

2.8.4 NIZK proofs

The NIWI proofs given above are not directly zero-knowledge, but in [GS12] the authors show under what conditions a proof system can be made composable zero-knowledge.

First, some restrictions are imposed to the set of quadratic equations. For all pairing product equations, the term on the right-hand side of the equation, $t_{\mathbb{T}}$, must be of the form $t_{\mathbb{T}} = \prod_{i=1}^{n} e(\hat{p}_i, \check{q}_i)$ for known $\hat{p}_i, \check{q}_i$.

In addition, the set of quadratic equations must be modified as follows. Let $\{\check{q}_j\}_{j=1}^{m}$ be all the constants that appear in the products of the right-hand sides of all pairing product equation, $(t_{\mathbb{T}})_k = \prod_{i=1}^{n} e(\hat{p}_{ik}, \check{q}_{ik})$. A set of variables $\{\check{z}_j\}_{j=1}^{m}$ and a set of equations $\{\check{z}_j - \check{q}_j = \hat{0}\}_{j=1}^{m}$ are added to the set of quadratic equations. In addition, in the pairing product equation the constants $\{\check{q}_j\}_{j=1}^{m}$ are substituted by the variables $\{\check{z}_j\}_{j=1}^{m}$. Note that the set of equations must be modified both when simulating proofs and when giving honest proofs created by the Prove algorithm.

To have zero-knowledge, it must exist an algorithm SimGenCRS that outputs a common reference string crs identically distributed to the one generated by WIGenCRS and, in addition, also outputs a trapdoor key tk. This trapdoor key must consist of two elements, $\mathbf{s}_{tk} \in \mathcal{R}^{\hat{m}\times 1}, \mathbf{r}_{tk} \in \mathcal{R}^{\hat{n}\times 1}$ such that $\iota_1'(1) = \iota_1'(0) + \mathbf{u}'\mathbf{s}_{tk}$ and $\iota_2'(1) = \iota_2'(0) + \mathbf{v}'\mathbf{r}_{tk}$, where $\iota_1', \mathbf{u}'$ are the inclusion map and the commitment keys for committing scalars to $\mathfrak{B}_1'$ and $\iota_2', \mathbf{v}'$ are the inclusion map and the commitment keys for committing scalars to $\mathfrak{B}_2'$.

The SimProve algorithm then works as follows. For all multiscalar multiplication equations in $\hat{\mathbb{G}}$ (resp. in $\check{\mathbb{H}}$), the right-hand side of the equation $\hat{t}$ (resp. $\check{t}$) is rewritten as $\delta\hat{t}$ (resp. $\delta\check{t}$) , where $\delta \in \mathbb{Z}_p$ is a new variable. In addition, for each quadratic equation in $\mathbb{Z}_p$, the right-hand side term of the equation t is rewritten as δt, where $\delta \in \mathbb{Z}_p$ is a new variable. Now all the rewritten equations have the trivial solution where all variables are 0 (in their respective groups). Therefore, the SimProve algorithm creates commitments to 0 for all variables. However, for the variables δ, the randomness for the commitments is not chosen at random; the commitments are created as $c_0 = \iota_1'(0) + \mathbf{u}'\mathbf{s}_{tk}$, $d_0 = \iota_2'(0) + \mathbf{v}'\mathbf{r}_{tk}$. Finally, proofs are created as with the Prove algorithm, but using these particular commitments.

In [GS12], the following theorem is stated and proven.

Theorem 2.8.2. *The proof system* (Setup, GenCRS, Prove, VerifyProof) *is a NIZK proof for satisfiability of a set of quadratic equations in bilinear groups with perfect completeness, perfect soundness and composable zero-knowledge if there exists algorithms* WIGenCRS, SimGenCRS, SimProve *such that* Setup, GenCRS, WIGenCRS, SimGenCRS, Prove, SimProve, VerifyProof *are built as specified above, the output generated by* GenCRS *is computationally indistinguishable from the output generated by* WIGenCRS *and the crs output by the* SimGenCRS *is identically distributed to the output of* WIGenCRS.

Once we have presented the Groth-Sahai proof framework it must be clear that, to instantiate it, only a few ingredients have to be specified. We must specify the algorithms Setup, GenCRS and WIGenCRS and we must proof that they satisfy the required properties. In addition, if instead of specifying the WIGenCRS algorithm we specify the SimGenCRS algorithm and we show that it satisfies the required conditions we will obtain zero-knowledge (for a reduced set of statements).

In some cases, however, it will be interesting to specify how proofs are created and verified since some non-out-of-the-box optimizations might be done. In the remaining of this section, we give the two most efficient constructions of the ones given in [GS12].

2.8.5 The SXDH instantiation

In [GS12], Groth and Sahai give one instantiation of Groth-Sahai proofs in asymmetric bilinear groups based on the SXDH assumption, which we now present. As already argued, we only need to specify the Setup, GenCRS and SimGenCRS algorithms to completely define the instantiation. However, we will also specify the Prove and VerifyProof algorithms both for sake of completeness and to show some optimizations which can be done on this instantiation.

Setup**:** on input a security parameter 1^λ, the setup outputs $gk := (p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathcal{PG}_{SXDH}(1^\lambda)^2$

GenCRS**:** on input gk, return the common reference string crs $:= (\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, \check{\mathbf{v}}_1, \check{\mathbf{v}}_2)$, where $\hat{\mathbf{u}}_1 = (\hat{g}, \alpha\hat{g})^\top \in \hat{\mathbb{G}}^2$, $\hat{\mathbf{u}}_2 = t_1\hat{\mathbf{u}}_1 \in \hat{\mathbb{G}}^2$, $\check{\mathbf{v}}_1 = (\check{h}, \beta\check{h})^\top \in \check{\mathbb{H}}^2$ and $\check{\mathbf{v}}_2 = t_2\check{\mathbf{v}}_1 \in \check{\mathbb{H}}^2$ for random $\alpha, \beta, t_1, t_2 \leftarrow \mathbb{Z}_p$. We will write $\hat{U} = (\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2) \in \hat{\mathbb{G}}^{2\times 2}$ and $\check{V} = (\check{\mathbf{v}}_1, \check{\mathbf{v}}_2) \in \check{\mathbb{H}}^{2\times 2}$.

No matrix H is needed as part of the common reference string. The modules $\mathfrak{B}_1, \mathfrak{B}_2, \mathfrak{B}_T$ are implicitly defined as $\hat{\mathbb{G}}^2, \check{\mathbb{H}}^2, \mathbb{T}^4$ respectively, and are used for all equation types. The required maps are defined as follows:

- The map $F : \hat{\mathbb{G}}^2 \times \check{\mathbb{H}}^2 \to \mathbb{T}^4$ is defined as:

$$F : \left(\begin{pmatrix} \hat{x}_1 \\ \hat{x}_2 \end{pmatrix}, \begin{pmatrix} \check{y}_1 \\ \check{y}_2 \end{pmatrix} \right) \mapsto \begin{pmatrix} e(\hat{x}_1, \check{y}_1) & e(\hat{x}_1, \check{y}_2) \\ e(\hat{x}_2, \check{y}_1) & e(\hat{x}_2, \check{y}_2) \end{pmatrix},$$

- The maps $\iota_1 : \hat{\mathbb{G}} \to \hat{\mathbb{G}}^2$ and $\mathfrak{p}_1 : \hat{\mathbb{G}}^2 \to \hat{\mathbb{G}}$ are defined as:

$$\iota_1(\hat{z}) := (\hat{0}, \hat{z})^\top, \qquad \mathfrak{p}_1((\hat{z}_1, \hat{z}_2)^\top) := \hat{z}_2 - \alpha\hat{z}_2$$

- The maps $\iota'_1 : \mathbb{Z}_p \to \hat{\mathbb{G}}^2$ and $\mathfrak{p}'_1 : \hat{\mathbb{G}}^2 \to \mathbb{Z}_p$ are defined as:

$$\iota'_1(z) := z(\hat{\mathbf{u}}_2 + (\hat{0}, \hat{g})^\top), \qquad \mathfrak{p}'_1((z_2\hat{g}, z_1\hat{g})^\top) := z_2 - \alpha z_1$$

- Maps $\iota_2, \iota'_2, \mathfrak{p}_2, \mathfrak{p}'_2$ are defined in an analogous way.

The maps $\iota_T, \mathfrak{p}_T$ are defined as follows for each equation type:

[2] We write $\mathcal{PG}_{SXDH}$ to say that the $SXDH$ assumption holds relative to $\mathcal{PG}$.

Pairing product equation: Commitment keys $\hat{U}$ and $\check{V}$ and inclusion maps ι_1 and ι_2 are used to commit to elements in $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$ respectively. The maps $\iota_T, \mathfrak{p}_T$ are defined as:

$$\iota_T(t_{\mathbb{T}}) := \begin{pmatrix} 0_{\mathbb{T}} & 0_{\mathbb{T}} \\ 0_{\mathbb{T}} & t_{\mathbb{T}} \end{pmatrix}, \quad \mathfrak{p}_T\left(\begin{pmatrix} e(\hat{x}_1, \check{y}_1) & e(\hat{x}_1, \check{y}_2) \\ e(\hat{x}_2, \check{y}_1) & e(\hat{x}_2, \check{y}_2) \end{pmatrix}\right) := e(\hat{x}_2 - \alpha\hat{x}_1, \check{y}_2 - \beta\check{y}_1)$$

Multiscalar multiplication in $\hat{\mathbb{G}}$: Commitment keys $\hat{U}$ and $\check{\mathbf{v}}_1$ and inclusion maps ι_1 and ι_2' are used to commit to elements in $\hat{\mathbb{G}}$ and $\mathbb{Z}_p$ respectively. The maps $\hat{\iota}_T, \hat{\mathfrak{p}}_T$ are defined as:

$$\hat{\iota}_T(\hat{z}) := F(\iota_1(\hat{z}), \iota_2'(1)), \qquad \hat{\mathfrak{p}}_T(Z_{\mathbb{T}}) := e^{-1,\hat{\mathbb{G}}}(\mathfrak{p}_T(Z_{\mathbb{T}}))$$

Multiscalar multiplication in $\check{\mathbb{H}}$: Commitment keys $\hat{\mathbf{u}}_1$ and $\check{V}$ and inclusion maps ι_1' and ι_2 are used to commit to elements in $\mathbb{Z}_p$ and $\check{\mathbb{H}}$ respectively. The maps $\check{\iota}_T, \check{\mathfrak{p}}_T$ are defined as:

$$\check{\iota}_T(\check{z}) := F(\iota_1'(1), \iota_2(\check{z})), \qquad \check{\mathfrak{p}}_T(Z_{\mathbb{T}}) := e^{-1,\check{\mathbb{H}}}(\mathfrak{p}_T(Z_{\mathbb{T}}))$$

Quadratic equations: Commitment keys $\hat{\mathbf{u}}_1$ and $\check{\mathbf{v}}_1$ and inclusion maps ι_1' and ι_2' are used to commit to elements in $\mathbb{Z}_p$. The maps $\iota_T', \mathfrak{p}_T'$ are defined as:

$$\iota_T' := F(\iota_1'(1), \iota_2'(t)), \qquad \mathfrak{p}_T'(Z_{\mathbb{T}}) := \log_{e(\hat{g},\check{h})} p_T(Z_{\mathbb{T}})$$

where $e^{-1,\hat{\mathbb{G}}}(e(\hat{z}, \check{h})) = \hat{z}$ and $e^{-1,\check{\mathbb{H}}}(e(\hat{g}, \check{z})) = \check{z}$.

SimGenCRS**:** on input gk, return the common reference string crs $:= (\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, \check{\mathbf{v}}_1, \check{\mathbf{v}}_2)$ and the trapdoor key $tk = (s_{tk}, r_{tk}) = (t_1, t_2)$, where $\hat{\mathbf{u}}_1 = (\hat{g}, \alpha\hat{g})^\top \in \hat{\mathbb{G}}^2$, $\hat{\mathbf{u}}_2 = t_1\hat{\mathbf{u}}_1 - (\hat{0}, \hat{g})^\top \in \hat{\mathbb{G}}^2$, $\check{\mathbf{v}}_1 = (\check{h}, \beta\check{h})^\top \in \check{\mathbb{H}}^2$ and $\check{\mathbf{v}}_2 = t_2\check{\mathbf{v}}_1 - (\check{0}, \check{h})^\top \in \check{\mathbb{H}}^2$ for random $\alpha, \beta, t_1, t_2 \leftarrow \mathbb{Z}_p$.

The required maps and matrices are implicitly defined exactly as in the GenCRS algorithm.

Prove**:** On input gk,crs, a set of equations and a witness $\hat{\mathbf{x}}, \check{\mathbf{y}}, \mathbf{x}, \mathbf{y}$ do:

1. Commit to the group elements $\hat{\mathbf{x}} \in \hat{\mathbb{G}}^m$ and the scalars $\mathbf{x} \in \mathbb{Z}_p^{m'}$ as

$$\hat{C} = \iota_1(\hat{\mathbf{x}}^\top) + \hat{U}R \in \hat{\mathbb{G}}^{2\times m}, \quad \hat{C}' = \iota_1(\mathbf{x}^\top) + \hat{\mathbf{u}}_1\mathbf{r} \in \hat{\mathbb{G}}^{2\times m'}, \quad \text{where } R \leftarrow \mathbb{Z}_p^{2\times m}, \mathbf{r} \leftarrow \mathbb{Z}_p^{1\times m'}$$

Commit to the group elements $\check{\mathbf{y}} \in \check{\mathbb{H}}^n$ and the scalars $\mathbf{y} \in \mathbb{Z}_p^{n'}$ as

$$\check{D} = \iota_2(\check{\mathbf{y}}^\top) + \check{V}S \in \check{\mathbb{H}}^{2\times n}, \quad \check{D}' = \iota_2(\mathbf{y}^\top) + \check{\mathbf{v}}_1\mathbf{s} \in \check{\mathbb{H}}^{2\times n'}, \quad \text{where } S \leftarrow \mathbb{Z}_p^{2\times n}, \mathbf{s} \leftarrow \mathbb{Z}_p^{1\times n'}$$

2. For each pairing product equation $\hat{\mathbf{a}} \cdot \check{\mathbf{y}} + \hat{\mathbf{x}} \cdot \check{\mathbf{b}} + \hat{\mathbf{x}} \cdot \Gamma\check{\mathbf{y}} = t_{\mathbb{T}}$ sample $T \leftarrow \mathbb{Z}_p^{2\times 2}$ and compute:

$$\check{\Pi} := \left(\iota_2(\check{\mathbf{b}}^\top) + \iota_2(\check{\mathbf{y}}^\top)\Gamma^\top + \check{V}S\Gamma^\top\right) R^\top - \check{V}T^\top \in \check{\mathbb{H}}^{2\times 2},$$
$$\hat{\Theta} := \left(\iota_1(\hat{\mathbf{a}}^\top) + \iota_1(\hat{\mathbf{x}}^\top)\Gamma\right) S^\top + \hat{U}T \in \hat{\mathbb{G}}^{2\times 2}.$$

For each linear equation $\hat{\mathbf{a}} \cdot \check{\mathbf{y}} = t_{\mathbb{T}}$ compute the proof as $\check{\Pi} := \check{0}$ and $\hat{\Theta} := \iota_1(\hat{\mathbf{a}}^\top)S^\top$. The proof can be communicated by just sending $\hat{\mathbf{a}}^\top S^\top$ since there is a bijective correspondence between $\iota_1(\hat{\mathbf{a}}^\top)S^\top$ and $\hat{\mathbf{a}}^\top S^\top$.

For each linear equation $\hat{\mathbf{x}} \cdot \check{\mathbf{b}} = t_{\mathbb{T}}$, the proof is computed as $\check{\Pi} := \iota_2(\check{\mathbf{b}}^\top)R^\top$, $\hat{\Theta} := \hat{0}$ and it can be communicated by sending $\check{\mathbf{b}}^\top R^\top$.

3. For each multiscalar multiplication equation in $\hat{\mathbb{G}}$ of the form $\hat{\mathbf{a}} \cdot \mathbf{y} + \hat{\mathbf{x}} \cdot \mathbf{b} + \hat{\mathbf{x}} \cdot \Gamma\mathbf{y} = \hat{t}$, set $T \leftarrow \mathbb{Z}_p^{2\times 1}$ and compute the proof as

$$\check{\Pi} := \left(\iota_2'(\mathbf{b}^\top) + \iota_2'(\mathbf{y}^\top)\Gamma^\top + \check{\mathbf{v}}_1\mathbf{s}\Gamma^\top\right) R^\top - \mathbf{v}_1 T^\top \in \check{\mathbb{H}}^{2\times 2}$$
$$\hat{\boldsymbol{\theta}} := \left(\iota_1(\hat{\mathbf{a}}^\top) + \iota_1(\hat{\mathbf{x}}^\top)\Gamma\right) \mathbf{s}^\top + \hat{U}T \in \hat{\mathbb{G}}^{2\times 1}.$$

For each linear equation $\hat{\mathbf{a}} \cdot \mathbf{y} = \hat{t}$ the proof is computed as $\check{\Pi} := \check{0}$, $\hat{\boldsymbol{\theta}} := \iota_1(\hat{\mathbf{a}}^\top)\mathbf{s}^\top$ and it can be communicated by sending $\hat{\mathbf{a}}^\top \mathbf{s}^\top$.

For each linear equation $\hat{\mathbf{x}} \cdot \mathbf{b} = \hat{t}$ the proof is computed as $\check{\Pi} := \iota_2'(\mathbf{b}^\top)R^\top, \hat{\boldsymbol{\theta}} := \hat{\mathbf{0}}$ and it can be communicated by sending $\mathbf{b}^\top R^\top$.

4. For each multiscalar multiplication equation in $\check{\mathbb{H}}$ of the form $\mathbf{a} \cdot \check{\mathbf{y}} + \mathbf{x} \cdot \check{\mathbf{b}} + \mathbf{x} \cdot \Gamma\check{\mathbf{y}} = \check{t}$ set $T \leftarrow \mathbb{Z}_p^{1\times 2}$ and compute the proof as

$$\check{\boldsymbol{\pi}} := \left(\iota_2(\check{\mathbf{b}}^\top) + \iota_2(\check{\mathbf{y}}^\top)\Gamma^\top + \check{V}S\Gamma^\top\right) \mathbf{r}^\top - \check{V}T^\top \in \check{\mathbb{H}}^{2\times 1},$$
$$\hat{\Theta} := \left(\iota_1'(\mathbf{a}^\top) + \iota_1'(\mathbf{x}^\top)\Gamma\right) S^\top + \hat{\mathbf{u}}_1 T \in \hat{\mathbb{G}}^{2\times 2}.$$

For each linear equation $\mathbf{a} \cdot \check{\mathbf{y}} = \check{t}$ the proof is computed as $\check{\boldsymbol{\pi}} := \check{\mathbf{0}}$, $\hat{\Theta} := \iota_1'(\mathbf{a}^\top)S^\top$ and it can be communicated by sending $\mathbf{a}^\top S^\top$.

For each linear equation $\mathbf{x} \cdot \check{\mathbf{b}} = \check{t}$ the proof is computed as $\check{\boldsymbol{\pi}} =: \iota_2(\check{\mathbf{b}}^\top)\mathbf{r}^\top, \hat{\Theta} := \hat{0}$ and it can be communicated by sending $\check{\mathbf{b}}^\top \mathbf{r}^\top$.

5. For each quadratic equation $\mathbf{a} \cdot \mathbf{y} + \mathbf{x} \cdot \mathbf{b} + \mathbf{x} \cdot \Gamma\mathbf{y} = t$, set $T \leftarrow \mathbb{Z}_p$ and compute the proof as

$$\check{\boldsymbol{\pi}} := \left(\iota_2'(\mathbf{b}^\top) + \iota_2'(\mathbf{y}^\top)\Gamma^\top + \check{\mathbf{v}}_1\mathbf{s}\Gamma^\top\right) \mathbf{r}^\top - \check{\mathbf{v}}_1 T \in \check{\mathbb{H}}^{2\times 1},$$
$$\hat{\boldsymbol{\theta}} := \left(\iota_1'(\mathbf{a}^\top) + \iota_1'(\mathbf{x}^\top)\Gamma\right) \mathbf{s}^\top + \hat{\mathbf{u}}_1 T \in \hat{\mathbb{G}}^{2\times 1}.$$

For each linear equation $\mathbf{a} \cdot \mathbf{y} = t$ the proof is computed as $\check{\boldsymbol{\pi}} := \check{\mathbf{0}}, \hat{\boldsymbol{\theta}} := \iota_1'(\mathbf{a}^\top)\mathbf{s}^\top$ and it can be communicated by sending $\mathbf{a}^\top \mathbf{s}^\top$.

For each linear equation $\mathbf{x} \cdot \mathbf{b} = t$ the proof is computed as $\check{\boldsymbol{\pi}} := \iota_2'(\mathbf{b}^\top)\mathbf{r}^\top, \hat{\boldsymbol{\theta}} := \hat{\mathbf{0}}$ and it can be communicated by sending $\mathbf{b}^\top \mathbf{r}^\top$.

VerifyProof: On input (gk, crs), a set of equations, and a proof $\hat{C}, \check{D}, \hat{C}', \check{D}', \{\check{\Pi}_i, \hat{\Theta}_i\}_{i=1}^N$ do:

1. For each pairing product equation with proof $(\check{\Pi}, \hat{\Theta})$ check that

$$\iota_1(\hat{\mathbf{a}}^\top) \bullet \check{D} + \hat{C} \bullet \iota_2(\check{\mathbf{b}}^\top) + \hat{C} \bullet \check{D}\Gamma^\top = \iota_T(t_{\mathbb{T}}) + \hat{U} \bullet \check{\Pi} + \hat{\Theta} \bullet \check{V}$$

Assumption: SXDH	$\hat{\mathbb{G}}$	$\check{\mathbb{H}}$	$\mathbb{Z}_p$
Variables $x \in \mathbb{Z}_p$, $\hat{x} \in \hat{\mathbb{G}}$	2	0	0
Variables $y \in \mathbb{Z}_p$, $\check{y} \in \check{\mathbb{H}}$	0	2	0

Figure 2.1: Cost of committing to each variable type.

Assumption: SXDH	$\hat{\mathbb{G}}$	$\check{\mathbb{H}}$	$\mathbb{Z}_p$
Pairing product equations	4	4	0
- Linear equation: $\hat{\mathbf{a}} \cdot \check{\mathbf{y}} = t_{\mathbb{T}}$	2	0	0
- Linear equation: $\hat{\mathbf{x}} \cdot \check{\mathbf{b}} = t_{\mathbb{T}}$	0	2	0
Multiscalar multiplication equations in $\hat{\mathbb{G}}$:	2	4	0
- Linear equation: $\hat{\mathbf{a}} \cdot \mathbf{y} = \hat{t}$	1	0	0
- Linear equation: $\hat{\mathbf{x}} \cdot \mathbf{b} = \hat{t}$	0	0	2
Multiscalar multiplication equations in $\check{\mathbb{H}}$:	4	2	0
- Linear equation: $\mathbf{a} \cdot \check{\mathbf{y}} = \check{t}$	0	1	0
- Linear equation: $\mathbf{x} \cdot \check{\mathbf{b}} = \check{t}$	0	0	2
Quadratic equations in $\mathbb{Z}_p$	2	2	0
- Linear equation: $\mathbf{a} \cdot \mathbf{y} = t$	0	0	1
- Linear equation: $\mathbf{x} \cdot \mathbf{b} = t$	0	0	1

Figure 2.2: Size of an NIZK proof for each type of equation.

2. For each multi scalar equation in $\hat{\mathbb{G}}$ with proof $(\check{\Pi}, \hat{\boldsymbol{\theta}})$ check that

$$\iota_1(\hat{\mathbf{a}}^\top) \bullet \check{D}' + \hat{C} \bullet \iota_2'(\mathbf{b}^\top) + \hat{C} \bullet \check{D}'\Gamma^\top = \hat{\iota}_T(\hat{t}) + \hat{U} \bullet \check{\Pi} + F(\hat{\boldsymbol{\theta}}, \check{\mathbf{v}}_1)$$

3. For each multi scalar equation in $\check{\mathbb{H}}$ with proof $(\check{\boldsymbol{\pi}}, \hat{\Theta})$ check that

$$\iota_1'(\mathbf{a}^\top) \bullet \check{D} + \hat{C}' \bullet \iota_2(\check{\mathbf{b}}^\top) + \hat{C}' \bullet \check{D}\Gamma^\top = \check{\iota}_T(\check{t}) + F(\hat{\mathbf{u}}_1, \check{\boldsymbol{\pi}}) + \hat{\Theta} \bullet \check{V}$$

4. For each quadratic equation with proof $(\check{\boldsymbol{\pi}}, \hat{\boldsymbol{\theta}})$ check that

$$\iota_1'(\mathbf{a}^\top) \bullet \check{D}' + \hat{C}' \bullet \iota_2'(\mathbf{b}^\top) + \hat{C}' \bullet \check{D}'\Gamma^\top = \iota_T'(t) + F(\hat{\mathbf{u}}_1, \check{\boldsymbol{\pi}}) + F(\hat{\boldsymbol{\theta}}, \check{\mathbf{v}}_1)$$

Size. Each element in $\mathfrak{B}_1$ and $\mathfrak{B}_2$ consists of two group elements from, respectively $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$. Therefore, the cost of each proof can be derived from the figures Fig. 2.1 and Fig. 2.2.

In [GS12], the authors show that the instantiation satisfies the requirements of the Groth-Sahai proof framework. This allows them to prove the following theorem.

Theorem 2.8.3. *The protocol given above is a NIWI proof with perfect completeness, perfect soundness and composable witness-indistinguishability for satisfiability of a set of equations over a bilinear group where the SXDH problem is hard. The protocol is zero-knowledge if for all pairing product equations, the term on the right-hand side of the equation,* $t_{\mathbb{T}}$*, is of the form* $t_{\mathbb{T}} = \prod_{i=1}^{n} e(\hat{p}_i, \check{q}_i)$ *for public* $\hat{p}_i, \check{q}_i$ *and the equations are rewritten so that they can be simulated.*

2.8.6 The DLIN instantiation

Another interesting instantiation of Groth-Sahai proofs given in [GS12], is the one based on the DLIN assumption for symmetric bilinear groups where $\hat{\mathbb{G}} = \check{\mathbb{H}} = \bar{\mathbb{G}}$. The interest of this instantiation is the introduction of Groth-Sahai proofs in the symmetric case and the fact that two maps F, F_{sym} are used to optimize the proofs efficiency.

Setup : $gk := (p, \bar{\mathbb{G}}, \mathbb{T}, e, \bar{g}) \leftarrow \mathcal{SPG}_{DLIN}(1^\lambda)$[3]

GenCRS : On input gk, return $\mathsf{crs} := (\bar{\mathbf{u}}_1, \bar{\mathbf{u}}_2, \bar{\mathbf{u}}_3)$, where $\bar{\mathbf{u}}_1 = (\alpha\bar{g}, \bar{0}, \bar{g})^\top, \bar{\mathbf{u}}_2 = (\bar{0}, \beta\bar{g}, \bar{g})^\top, \bar{\mathbf{u}}_3 = r\bar{\mathbf{u}}_1 + s\bar{\mathbf{u}}_2$ for random $\alpha, \beta \leftarrow \mathbb{Z}_p^*$ and $r, s \leftarrow \mathbb{Z}_p$. Let us define $\bar{U} = (\bar{\mathbf{u}}_1, \bar{\mathbf{u}}_2, \bar{\mathbf{u}}_3) \in \bar{\mathbb{G}}^{3\times 3}$ and $\bar{U'} = (\bar{\mathbf{u}}_1, \bar{\mathbf{u}}_2) \in \bar{\mathbb{G}}^{3\times 2}$.

The modules $\mathfrak{B}_1, \mathfrak{B}_2, \mathfrak{B}_T$ are implicitly defined as $\bar{\mathbb{G}}^3, \bar{\mathbb{G}}^3, \mathbb{T}^9$ respectively, and are used for all equation types. The maps are defined as follows:

- Two different bilinear maps F, F_{sym} are defined. The map $F : \bar{\mathbb{G}}^3 \times \bar{\mathbb{G}}^3 \to \mathbb{T}^9$ is defined as

$$F : \left(\begin{pmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \bar{x}_3 \end{pmatrix}, \begin{pmatrix} \bar{y}_1 \\ \bar{y}_2 \\ \bar{y}_3 \end{pmatrix} \right) \mapsto \begin{pmatrix} e(\bar{x}_1, \bar{y}_1) & e(\bar{x}_1, \bar{y}_2) & e(\bar{x}_1, \bar{y}_3) \\ e(\bar{x}_2, \bar{y}_1) & e(\bar{x}_2, \bar{y}_2) & e(\bar{x}_2, \bar{y}_3) \\ e(\bar{x}_3, \bar{y}_1) & e(\bar{x}_3, \bar{y}_2) & e(\bar{x}_3, \bar{y}_3) \end{pmatrix},$$

  whereas the symmetric map F_{sym} is defined by

$$F_{sym}(\bar{\mathbf{x}}, \bar{\mathbf{y}}) := \frac{1}{2}\tilde{F}(\bar{\mathbf{x}}, \bar{\mathbf{y}}) + \frac{1}{2}\tilde{F}(\bar{\mathbf{y}}, \bar{\mathbf{x}})$$

- The maps $\iota : \bar{\mathbb{G}} \to \bar{\mathbb{G}}^3$ and $\mathfrak{p} : \bar{\mathbb{G}}^3 \to \bar{\mathbb{G}}$ are defined as:

$$\iota(\bar{z}) := (\bar{0}, \bar{0}, \bar{z})^\top, \qquad \mathfrak{p}((\bar{z}_1, \bar{z}_2, \bar{z}_3)^\top) = \bar{z}_3 - \frac{1}{\alpha}\bar{z}_1 - \frac{1}{\beta}\bar{z}_2$$

- The maps $\iota' : \mathbb{Z}_p \to \bar{\mathbb{G}}^3$ and $\mathfrak{p}' : \bar{\mathbb{G}}^3 \to \mathbb{Z}_p$ are defined as:

$$\iota'(z) = z(\bar{\mathbf{u}}_3 + (\bar{0}, \bar{0}, \bar{g})^\top), \qquad \mathfrak{p}'((z_1\bar{g}, z_2\bar{g}, z_3\bar{g})^\top) = z_3 - \frac{1}{\alpha}z_1 - \frac{1}{\beta}z_2$$

The matrices $\{H_i\}$ and the maps $\iota_T, \mathfrak{p}_T$ are defined as follows for each equation type:

[3] We write $\mathcal{SPG}_{DLIN}$ to say that the $DLIN$ assumption holds relative to $\mathcal{SPG}$.

Pairing product equation: Commitment keys $\bar{U}$ and the inclusion map ι are used to commit to elements in $\bar{\mathbb{G}}$. The maps $\iota_T, \mathfrak{p}_T$ are the same for both F and F_{sym}:

$$\iota_T(t_{\mathbb{T}}) := \begin{pmatrix} 0_{\mathbb{T}} & 0_{\mathbb{T}} & 0_{\mathbb{T}} \\ 0_{\mathbb{T}} & 0_{\mathbb{T}} & 0_{\mathbb{T}} \\ 0_{\mathbb{T}} & 0_{\mathbb{T}} & t_{\mathbb{T}} \end{pmatrix},$$

$$\begin{aligned} \mathfrak{p}_T(Z_{\mathbb{T}}) :=& z_{33,\mathbb{T}} - \alpha^{-1} z_{13,\mathbb{T}} - \beta^{-1} z_{23,\mathbb{T}} \\ & - \alpha^{-1}(z_{31,\mathbb{T}} - \alpha^{-1} z_{11,\mathbb{T}} - \beta^{-1} z_{21,\mathbb{T}}) \\ & - \beta^{-1}(z_{32,\mathbb{T}} - \alpha^{-1} z_{12,\mathbb{T}} - \beta^{-1} z_{22,\mathbb{T}}), \end{aligned}$$

where $Z_{\mathbb{T}} = \begin{pmatrix} z_{11,\mathbb{T}} & z_{12,\mathbb{T}} & z_{13,\mathbb{T}} \\ z_{21,\mathbb{T}} & z_{22,\mathbb{T}} & z_{23,\mathbb{T}} \\ z_{31,\mathbb{T}} & z_{32,\mathbb{T}} & z_{33,\mathbb{T}} \end{pmatrix}$.

The matrices

$$H_1 = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \qquad H_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}, \qquad H_3 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}$$

are a basis of all matrices H such that $\bar{U}H \bullet_{sym} \bar{U} = \bar{0}$ whereas the equation $\bar{U}H \bullet \bar{U} = \bar{0}$ only admits the trivial solution.

Multiscalar multiplication equations: Commitment keys $\bar{U}$ and $\bar{U}'$ and inclusion maps ι and ι' are used to commit to elements in $\bar{\mathbb{G}}$ and $\mathbb{Z}_p$ respectively. Both for $\tilde{F} = F$ and $\tilde{F} = F_{sym}$, $\tilde{\bar{\iota}}_T$, $\tilde{\bar{\mathfrak{p}}}_T$ are defined as:

$$\tilde{\bar{\iota}}_T(\bar{z}) := \tilde{F}(\iota'(1), \iota(\bar{z})), \qquad \tilde{\bar{\mathfrak{p}}}_T(Z_{\mathbb{T}}) := e^{-1}(\mathfrak{p}_T(Z_{\mathbb{T}}))$$

where $e^{-1}(e(\bar{z}, \bar{g})) = \bar{z}$.

The matrix

$$H_1 = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

generates all matrices H such that $\bar{U}'H \bullet_{sym} \bar{U} = \bar{0}$ whereas the equation $\bar{U}'H \bullet \bar{U} = \bar{0}$ only admits the trivial solution.

Quadratic equations: Commitment keys $\bar{U}'$ and the inclusion map ι' are used to commit to elements in $\mathbb{Z}_p$. Both for $\tilde{F} = F$ and $\tilde{F} = F_{sym}$, $\tilde{\iota}'_T$, $\tilde{\mathfrak{p}}'_T$ are defined as:

$$\tilde{\iota}'_T(z) := \tilde{F}(\iota'(1), \iota'(z)), \qquad \tilde{\mathfrak{p}}'_T(Z_{\mathbb{T}}) := \log_{e(\bar{g},\bar{g})}(\mathfrak{p}_T(Z_{\mathbb{T}}))$$

The matrix

$$H_1 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

generates all matrices H such that $\bar{U}'H \bullet_{sym} \bar{U}' = \bar{0}$ whereas the equation $\bar{U}'H \bullet \bar{U}' = \bar{0}$ only admits the trivial solution.

$\mathsf{SimGenCRS}$: On input gk, return $\mathsf{crs} := (\bar{\mathbf{u}}_1, \bar{\mathbf{u}}_2, \bar{\mathbf{u}}_3)$ and the trapdoor key $tk = \mathbf{s}_{tk} = (r, s)^\top$, where $\bar{\mathbf{u}}_1 = (\alpha\bar{g}, \bar{0}, \bar{g})^\top, \bar{\mathbf{u}}_2 = (\bar{0}, \beta\bar{g}, \bar{g})^\top, \bar{\mathbf{u}}_3 = r\bar{\mathbf{u}}_1 + s\bar{\mathbf{u}}_2 - (\bar{0}, \bar{0}, \bar{g})^\top$ for random $\alpha, \beta \leftarrow \mathbb{Z}_p^*$ and $r, s \leftarrow \mathbb{Z}_p$.

The required maps and matrices are implicitly defined exactly as in the GenCRS algorithm.

Prove : On input gk,crs, a set of equations and a witness $\bar{\mathbf{x}}, \mathbf{x}$ do:

1. Commit to the scalars $\mathbf{x} \in \mathbb{Z}_p^{m'}$ and the group elements $\bar{\mathbf{x}} \in \bar{\mathbb{G}}^m$ as

   $$\bar{C}' := \iota'(\mathbf{x}^\top) + \bar{U}'R \in \bar{\mathbb{G}}^{3\times m'}, \quad \bar{C} := \iota(\bar{\mathbf{x}}^\top) + \bar{U}S \in \bar{\mathbb{G}}^{3\times m}, \quad \text{where } R \leftarrow \mathbb{Z}_p^{2\times m'}, S \leftarrow \mathbb{Z}_p^{3\times m}$$

2. For each pairing product equation $\bar{\mathbf{a}} \cdot \bar{\mathbf{x}} + \bar{\mathbf{x}} \cdot \Gamma\bar{\mathbf{x}} = t_{\mathbb{T}}$ the map F_{sym} is used, random $r_1, r_2, r_3 \leftarrow \mathbb{Z}_p$ are sampled and the proof is computed as:

   $$\bar{\Phi} := \iota(\bar{\mathbf{a}}^\top)S^\top + \iota(\bar{\mathbf{x}}^\top)(\Gamma + \Gamma^\top)S^\top + \bar{U}(S\Gamma^\top S^\top + \sum_{i=1}^{3} r_i H_i)$$

   For each linear equation $\bar{\mathbf{a}} \cdot \bar{\mathbf{x}} = t_{\mathbb{T}}$, the map F is used and the proof is computed as $\bar{\Pi} := \bar{0}$ and $\bar{\Theta} := \iota(\bar{\mathbf{a}}^\top)S^\top$. The proof can be communicated by just sending $\bar{\Phi} := \bar{\mathbf{a}}^\top S^\top$.

3. For each multiscalar multiplication equation $\mathbf{a} \cdot \bar{\mathbf{x}} + \mathbf{x} \cdot \bar{\mathbf{a}} + \mathbf{x} \cdot \Gamma\bar{\mathbf{x}} = \bar{t}$ the map F_{sym} is used and a random $r_1 \leftarrow \mathbb{Z}_p$ is sampled to compute the proof as

   $$\bar{\Phi} := \iota(\bar{\mathbf{a}}^\top)R^\top + \iota(\bar{\mathbf{x}}^\top)\Gamma^\top R^\top + \iota'(\mathbf{a}^\top)S^\top + \iota'(\mathbf{x}^\top)\Gamma S^\top + \bar{U}(S\Gamma^\top R^\top + r_1 H_1^\top)$$

   For each linear equation $\mathbf{a} \cdot \bar{\mathbf{x}} = \bar{t}$ the map F is used and the proof is computed as $\bar{\Pi} := \bar{0}$ and $\bar{\Theta} := \iota'(\mathbf{a}^\top)S^\top$. The proof can be communicated by just sending $\Phi := \mathbf{a}^\top S^\top$.

   For each linear equation $\mathbf{x} \cdot \bar{\mathbf{a}} = \bar{t}$ the map F is used and the proof is computed as $\bar{\Pi} := \iota(\bar{\mathbf{a}}^\top)R^\top, \bar{\Theta} := \bar{0}$. The proof can be communicated by sending $\bar{\Phi} := \bar{\mathbf{a}}^\top R^\top$.

4. For each quadratic equation $\mathbf{a} \cdot \mathbf{x} + \mathbf{x} \cdot \Gamma\mathbf{x} = t$ the map F_{sym} is used, a random $r_1 \leftarrow \mathbb{Z}_p$ is sampled and the proof is computed as:

   $$\bar{\Phi} := \iota'(\mathbf{a}^\top)R^\top + \iota'(\mathbf{x}^\top)(\Gamma + \Gamma^\top)R^\top + \bar{U}'(R\Gamma^\top R^\top + r_1 H_1)$$

   For each linear equation $\mathbf{a} \cdot \mathbf{x} = t$, the map F is used and the proof is computed as $\bar{\Pi} := \bar{0}$ and $\bar{\Theta} := \iota'(\mathbf{a}^\top)R^\top$. The proof can be communicated by just sending $\Phi := \mathbf{a}^\top R^\top$.

Assumption: DLIN	$\bar{\mathbb{G}}$	$\mathbb{Z}_p$
Variables $x \in \mathbb{Z}_p$, $\bar{x} \in \bar{\mathbb{G}}$	3	0

Figure 2.3: Cost of committing to each variable type.

$\mathsf{VerifyProof}$. On input (gk, crs), a set of equations, and a proof $\bar{C}, \bar{C}', \{\bar{\Phi}_i\}_{i=1}^N$ do:

1. For each pairing product equation with proof $\bar{\Phi}$ check that

$$\iota(\bar{\mathbf{a}}^\top) \bullet_{sym} \bar{C} + \bar{C} \bullet_{sym} \bar{C}\Gamma^\top = \iota_T(t_{\mathbb{T}}) + \bar{U} \bullet_{sym} \bar{\Phi}$$

   For each linear equation with proof $\bar{\Phi}$ check that

$$\iota(\bar{\mathbf{a}}^\top) \bullet \bar{C} = \iota_T(t_{\mathbb{T}}) + \iota(\bar{\Phi}) \bullet \bar{U}$$

2. For each multi scalar multiplication equation proof $\bar{\Phi}$ check that

$$\iota'(\mathbf{a}^\top) \bullet_{sym} \bar{C} + \bar{C}' \bullet_{sym} \iota(\bar{\mathbf{a}}^\top) + \bar{C}' \bullet_{sym} \bar{C}\Gamma^\top = \bar{\iota}_{sym,T}(\bar{t}) + \bar{U} \bullet_{sym} \bar{\Phi}$$

   For each linear equation with proof Φ check that

$$\iota'(\mathbf{a}^\top) \bullet \bar{C} = \bar{\iota}_T(\bar{t}) + \iota'(\Phi) \bullet \bar{U}$$

   For each linear equation with proof $\bar{\Phi}$ check that

$$\bar{C}' \bullet \iota(\bar{\mathbf{a}}^\top) = \bar{\iota}_T(\bar{t}) + \bar{U}' \bullet \iota(\bar{\Phi})$$

3. For each quadratic equation with proof $\bar{\Phi}$ check that

$$\bar{C}' \bullet_{sym} \iota'(\mathbf{a}^\top) + \bar{C}' \bullet_{sym} \bar{C}'\Gamma^\top = \iota'_{sym,T}(t) + \bar{U}' \bullet_{sym} \bar{\Phi}$$

   For each linear equation with proof Φ check that

$$\bar{C}' \bullet \iota'(\mathbf{a}^\top) = \iota'_T(t) + \bar{U}' \bullet \iota'(\Phi)$$

Size. The cost of each proof can be derived from the figures Fig. 2.3 and Fig. 2.4.

In [GS12], they show that the instantiation satisfies the requirements of the Groth-Sahai proof framework. This allows them to prove the following theorem.

Theorem 2.8.4. *The protocol given above is a NIWI proof with perfect completeness, perfect soundness and composable witness-indistinguishability for satisfiability of a set of equations over a bilinear group where the DLIN problem is hard. The protocol is zero-knowledge if for all pairing product equations, the term on the right-hand side of the equation, $t_{\mathbb{T}}$, is of the form $t_{\mathbb{T}} = \prod_{i=1}^n e(\bar{p}_i, \bar{q}_i)$ for public $\bar{p}_i, \bar{q}_i$ and the equations are rewritten so that they can be simulated.*

Assumption: DLIN	$\bar{\mathbb{G}}$	$\mathbb{Z}_p$
Pairing product equations	9	0
- Linear equation: $\bar{\mathbf{a}} \cdot \bar{\mathbf{y}} = t_{\mathbb{T}}$	3	0
Multiscalar multiplication equations:	9	0
- Linear equation: $\mathbf{a} \cdot \bar{\mathbf{y}} = \bar{t}$	0	3
- Linear equation: $\mathbf{x} \cdot \bar{\mathbf{b}} = \bar{t}$	2	0
Quadratic equations in $\mathbb{Z}_p$	6	0
- Linear equation: $\mathbf{x} \cdot \mathbf{b} = t$	0	2

Figure 2.4: Size of an NIZK proof for each type of equation.

Extensions of Groth-Sahai proofs

There have been several papers that extend or improve the Groth-Sahai proof system in different directions. [Mei09] suggested how to create perfectly extractable commitments, something which is not given by the commitments used by Groth and Sahai. [CHP12, BFI$^+$10] reduced the computational cost of the verification of the proofs using batch techniques, at the cost of trading perfect soundness for statistical soundness. [Seo12] gave another map for verifying proofs in the symmetric setting which reduces the computational cost of the verification of the proofs. On the other hand, they prove that the map proposed by Groth and Sahai in the asymmetric setting is optimal. [GSW10] proposed another assumption on which Groth-Sahai proofs can be based. [BCKL08, BCC$^+$09] exploited re-randomization properties of Groth-Sahai proofs, which they used in anonymous credentials. Recently, [CKLM12] introduced a new notion of malleable proof systems, which can be built from Groth-Sahai proofs. While there has been significant research effort devoted to pairing-based NIZK proofs, Groth-Sahai proofs still remain the most efficient NIZK proofs that are based on standard intractability assumptions and there has not been any progress in reducing their size or the prover's computation.

Chapter 3

An Algebraic Framework for Diffie-Hellman Assumptions

This chapter is mainly based on the results published in [EHK$^+$13] (conference version) and [EHK$^+$17] (journal version), coauthored with Gottfried Herold, Eike Kiltz, Carla Ràfols and Jorge Luis Villar.

3.1 Motivation

In Section 2.2, we have introduced the DDH assumption and we have discussed that the DDH problem is easy in a group $\bar{\mathbb{G}}$ if there exists a bilinear map $e : \bar{\mathbb{G}} \times \bar{\mathbb{G}} \to \mathbb{T}$. We have also mentioned the DLIN assumption, which can hold in symmetric bilinear groups since the DLIN problem can not be solved simply with a bilinear map.

A similar argument to the one used to show that the DDH problem is easy in symmetric bilinear groups can be used to show that if a 3-linear map $e : \bar{\mathbb{G}} \times \bar{\mathbb{G}} \times \bar{\mathbb{G}} \to \mathbb{T}_3$ existed then the DLIN problem would be trivially easy in $\bar{\mathbb{G}}$.

As a consequence, the DLIN assumption (also known as the $2-\mathsf{Lin}$ assumption) was generalized to the $(k-\mathsf{Lin})_{k\in\mathbb{N}}$ assumption family [HK07, Sha07]. Each $k-\mathsf{Lin}$ assumption does not hold in a group with a $k+1$-linear map, creating a family of increasingly weaker assumptions. This generalization also includes the DDH assumption, which corresponds to the $1-\mathsf{Lin}$ assumption.

The $k-\mathsf{Lin}$ assumptions can be seen as subgroup membership assumptions, i.e., assumptions stating that it is hard to say whether an element of a group is in a strict subgroup or not. For example, the DDH can be thought as the problem of distinguishing whether a pair of group elements $(\bar{r}, \bar{z}) \in \bar{\mathbb{G}}^2$ is sampled uniformly at random from $\bar{\mathbb{G}}^2$ or is in the subspace of $\bar{\mathbb{G}}^2$ generated by $(\bar{g}, a\bar{g}) \in \bar{\mathbb{G}}^2$ for some $a \leftarrow \mathbb{Z}_p$. Similarly, the $2-\mathsf{Lin}$ assumption can be seen as distinguishing whether a triplet of elements $(\bar{r}, \bar{s}, \bar{t})$ is sampled uniformly at random from $\bar{\mathbb{G}}^3$ or if it is in the subspace generated by the triplets $\{(a_1\bar{g}, \bar{0}, \bar{g}), (\bar{0}, a_2\bar{g}, \bar{g})\} \in \bar{\mathbb{G}}^3$ for $a_1, a_2 \leftarrow \bar{\mathbb{G}}$.

Formulating the $k-\mathsf{Lin}$ assumptions as subgroup membership assumptions has proven to be useful. For instance, it allowed to provide more instantiations of the original DDH-based scheme of Cramer and Shoup and it is also the most natural point of view for translating schemes originally constructed in composite order groups into prime order groups [Fre10, MSF10, Seo12, SC12]. Another number of works have illustrated the usefulness of a more algebraic point of view on decisional assumptions in bilinear groups, like the Dual Pairing Vector Spaces of Okamoto and Takashima [OT10] or the Subspace Assumption of Lewko [LOS+10]. Although these new decisional assumptions reduce to the 2-Lin Assumption, their flexibility and their algebraic description have proven to be crucial in many works to obtain complex primitives in strong security models previously unrealized in the literature, like Attribute-Based Encryption, Unbounded Inner Product Encryption and many more (see [LOS+10, OT15, OT12], just to name a few).

Even though the $k-\mathsf{Lin}$ assumptions have been the default assumptions used in most of the cryptographic constructions, in [EHK+17] we question whether there could be other assumption families eventually leading to more efficient cryptographic constructions. In particular, we look at assumptions formulated as subspace membership problems: to distinguish whether a given vector in $\bar{\mathbb{G}}^{\ell}$ is contained in the space spanned by k vectors in $\bar{\mathbb{G}}^{\ell}$.

3.2 Our contributions

We define a new framework for giving subspace membership assumptions, which we call DDH-like assumptions.

- We give three assumption families based on our framework, which in particular can be used as assumptions in bilinear groups.
- We give security reductions between these assumptions, as well as some results on the security of any assumption derived from our framework.
- We show how our assumptions can be used to build cryptographic protocols. In particular, we instantiate NIZK proofs from the Groth-Sahai framework based on our new assumptions.
- We give more efficient constructions for NIZK proofs for languages of statements related to the common reference string.

The assumptions' security results given in [EHK+17] were not part of my contribution and fall out of the scope of this thesis. Regarding cryptographic protocols, my main contribution was on applying our framework to NIZK proofs. Therefore, in this thesis we will only give the applications of our framework for NIZK proofs. We refer the reader to [EHK+17] for those results not detailed in this thesis.

A new framework for DDH-like assumptions

For integers $\ell > k$ let $\mathcal{D}_{\ell,k}$ be an (efficiently sampleable) distribution over $\mathbb{Z}_p^{\ell \times k}$. We define the $\mathcal{D}_{\ell,k}$-Matrix DH ($\mathcal{D}_{\ell,k} - \mathsf{MDDH}$) Assumption as the following subgroup decision assumption:

$$\mathcal{D}_{\ell,k} - \mathsf{MDDH} : \qquad (A||A\mathbf{r})\bar{g} \approx (A||\mathbf{u})\bar{g} \in \bar{\mathbb{G}}^{\ell \times (k+1)},$$

where $A \in \mathbb{Z}_p^{\ell \times k}$ is chosen from the distribution $\mathcal{D}_{\ell,k}$, $\mathbf{r} \leftarrow \mathbb{Z}_p^k$, and $\mathbf{u} \leftarrow \mathbb{Z}_p^\ell$.

New Assumptions for bilinear groups

We propose other families of decisional assumptions that did not previously appear in the literature, e.g., those associated to $\mathcal{C}_k, \mathcal{SC}_k, \mathcal{IL}_k$ defined below. For the most important parameters $k = 2$ and $\ell = k+1 = 3$, we consider the following examples of distributions:

$$\mathcal{C}_2 : A = \begin{pmatrix} a_1 & 0 \\ 1 & a_2 \\ 0 & 1 \end{pmatrix}, \ \mathcal{SC}_2 : A = \begin{pmatrix} a & 0 \\ 1 & a \\ 0 & 1 \end{pmatrix}, \ \mathcal{L}_2 : A = \begin{pmatrix} a_1 & 0 \\ 0 & a_2 \\ 1 & 1 \end{pmatrix}, \ \mathcal{IL}_2 : A = \begin{pmatrix} a & 0 \\ 0 & a+1 \\ 1 & 1 \end{pmatrix},$$

for uniform $a, a_1, a_2 \in \mathbb{Z}_p$ as well as $\mathcal{U}_{3,2}$, the uniform distribution in $\mathbb{Z}_p^{3 \times 2}$, already considered in [BHHO08, NS12, GHV12, Vil12]). It is easy to verify that $\mathcal{L}_2 - \mathsf{MDDH} = 2 - \mathsf{Lin}$. We define $2 - \mathsf{Casc} := \mathcal{C}_2 - \mathsf{MDDH}$ (Cascade Assumption), $2 - \mathsf{SCasc} := \mathcal{SC}_2 - \mathsf{MDDH}$ (Symmetric Cascade Assumption), and $2 - \mathsf{ILin} := \mathcal{IL}_2 - \mathsf{MDDH}$ (Incremental Linear Assumption). In [EHK$^+$17], we show that $2 - \mathsf{SCasc} \implies 2 - \mathsf{Casc}$, $2 - \mathsf{ILin} \implies 2 - \mathsf{Lin}$ and that $\mathcal{U}_{3,2} - \mathsf{MDDH}$ is the weakest of these assumptions. Although originally [EHK$^+$13] $2 - \mathsf{ILin}$ and $2 - \mathsf{SCasc}$ were thought to be incomparable assumptions, in [EHK$^+$17] we show that $2 - \mathsf{SCasc}$ and $2 - \mathsf{ILin}$ are indeed equivalent assumptions. The equivalence results, together with the fact that $2 - \mathsf{ILin} \implies 2 - \mathsf{Lin}$, imply that $2 - \mathsf{SCasc}$ is a stronger assumption than $2 - \mathsf{Lin}$.

Efficiency improvements

As a measure of efficiency, we define the representation size $\mathsf{RE}_{\bar{\mathbb{G}}}(\mathcal{D}_{\ell,k})$ of an $\mathcal{D}_{\ell,k} - \mathsf{MDDH}$ assumption as the minimal number of group elements needed to represent $A\bar{g}$ for any $A \in \mathcal{D}_{\ell,k}$. This parameter is important since it affects the performance (typically the size of public/secret parameters) of schemes based on a Matrix Diffie-Hellman Assumption. $2 - \mathsf{Lin}$ and $2 - \mathsf{Casc}$ have representation size 2 (elements $(a_1\bar{g}, a_2\bar{g})$), while $2 - \mathsf{SCasc}$'s representation size is only 1 (element $a\bar{g}$). Hence our new assumptions directly translate into shorter parameters for a large number of applications. Further, our result points out a trade-off between efficiency and hardness which questions the role of $2 - \mathsf{Lin}$ as the "standard decisional assumption" over a bilinear group $\bar{\mathbb{G}}$.

New Families of Weaker Assumptions

By defining appropriate distributions $\mathcal{C}_k, \mathcal{SC}_k, \mathcal{IL}_k$ over $\mathbb{Z}_p^{(k+1)\times k}$, for any $k \in \mathbb{N}$, one can generalize all three new assumptions naturally to $k - \mathsf{Casc}$, $k - \mathsf{SCasc}$, and $k - \mathsf{ILin}$ with representation size $k, 1$, and 1, respectively. In [EHK$^+$17] we show that $k - \mathsf{SCasc}$, and $k - \mathsf{ILin}$ are equivalent for every k. Since all these assumptions are false in $(k+1)$-linear groups this gives us three new families of increasingly strictly weaker assumptions[1]. In particular, the $k - \mathsf{SCasc}$ (equivalently, $k - \mathsf{ILin}$) assumption family is of great interest due to its compact representation size of only 1 element.

Groth-Sahai Non-Interactive Zero-Knowledge Proofs

We show how to instantiate the Groth-Sahai proof system based on any $\mathcal{D}_{\ell,k} - \mathsf{MDDH}$ Assumption. While the size of the proofs depends only on ℓ and k, the crs and verification depends on the representation size of the Matrix Assumptions. Therefore, our new instantiations offer improved efficiency over the $2 - \mathsf{Lin}$-based construction from [GS12]. This application in particular highlights the usefulness of the Matrix Assumption to describe in a compact way many instantiations of a scheme: instead of having to specify the constructions for the DDH and the $2 - \mathsf{Lin}$ assumptions separately [GS12], we can recover them as a special case of a general construction.

More efficient proofs for crs dependent languages

We provide more efficient NIZK proofs for concrete natural languages which are dependent on the common reference string. More specifically, the common reference string of our $\mathcal{D}_{\ell,k} - \mathsf{MDDH}$ instantiation of Groth-Sahai proofs of includes as part of the commitment keys the matrix $A\bar{g}$, where $A \leftarrow \mathcal{D}_{\ell,k}$. We give more efficient proofs for several languages related to A. Although at first glance the languages considered may seem quite restricted, they naturally appear in many applications, where typically $A\bar{g}$ is the public key of some encryption scheme and one wants to prove statements about ciphertexts. More specifically, we obtain improvements for several kinds of statements, namely:

- **Subgroup membership proofs.** We give more efficient proofs in the language $\mathcal{L}_{A,\bar{\mathbb{G}},\bar{g}} := \{A\mathbf{r}\bar{g}, \mathbf{r} \in \mathbb{Z}_p^k\} \subset \bar{\mathbb{G}}^\ell$. In particular, these proofs can be used to prove that two commitments with the same key hide the same value.
- **Ciphertext validity.** The result is extended to prove membership in the language $\mathcal{L}_{A,\mathbf{z},\bar{\mathbb{G}},\bar{g}} := \{\bar{\mathbf{c}} : \bar{\mathbf{c}} = A\mathbf{r}\bar{g} + \mathbf{z}\bar{m}\} \subset \bar{\mathbb{G}}^\ell$, where $\mathbf{z} \in \mathbb{Z}_p^\ell$ is some public vector such that $\mathbf{z} \notin \mathsf{Im}(A)$, and the witness of

[1] We actually assume that k and ℓ are considered as constants, i.e. they do not depend on the security parameter. Otherwise, for a general $\mathcal{D}_{\ell,k}$ it is not so easy to solve the $\mathcal{D}_{\ell,k} - \mathsf{MDDH}$ problem with the only help of a $(k+1)$-linear map, because determinants of size $k+1$ could not be computable in polynomial time.

the statement is $(\mathbf{r}, \bar{m}) \in \mathbb{Z}_p^k \times \bar{\mathbb{G}}$. The natural application of this result is to prove that a ciphertext is well-formed and the prover knows the message $\bar{m}$, like for instance in [DHLW10].

- **Plaintext equality.** We obtain more efficient proofs for equality of ciphertexts. We consider Groth-Sahai proofs in a setting in which the variables of the proofs are committed with different commitment keys, defined by two matrices $A \leftarrow \mathcal{D}_{\ell_1,k_1}, B \leftarrow \mathcal{D}'_{\ell_2,k_2}$. We give more efficient proofs of membership in the language $\mathcal{L}_{A,B,\bar{\mathbb{G}},\bar{g}} := \{(\bar{\mathbf{c}}_A, \bar{\mathbf{c}}_B) : \bar{\mathbf{c}}_A = A\mathbf{r}\bar{g} + (\bar{0}, \ldots, \bar{0}, \bar{m})^\top, \bar{\mathbf{c}}_B = B\mathbf{s}\bar{g} + (\bar{0}, \ldots, \bar{0}, \bar{m})^\top, \mathbf{r} \in \mathbb{Z}_p^{k_1}, \mathbf{s} \in \mathbb{Z}_p^{k_2}\} \subset \bar{\mathbb{G}}^{\ell_1} \times \bar{\mathbb{G}}^{\ell_2}$ As in the previous case, this language appears most naturally when one wants to prove equality of two committed values or plaintexts encrypted under different keys.

3.3 Matrix DH assumptions

Definition 3.3.1. *Let $\ell, k \in \mathbb{N}$ with $\ell > k$. We call $\mathcal{D}_{\ell,k}$ a matrix distribution if it outputs (in poly time, with overwhelming probability) matrices in $\mathbb{Z}_p^{\ell \times k}$ of full rank k. We define $\mathcal{D}_k := \mathcal{D}_{k+1,k}$.*

For simplicity, we will also assume that, w.l.o.g., the first k rows of $A \leftarrow \mathcal{D}_{\ell,k}$ form an invertible matrix.

We define the $\mathcal{D}_{\ell,k}$-matrix problem as the problem of distinguishing the two distributions $(A\bar{g}, A\mathbf{w}\bar{g})$ and $(A\bar{g}, \mathbf{u}\bar{g})$, where $A \leftarrow \mathcal{D}_{\ell,k}, \mathbf{w} \leftarrow \mathbb{Z}_p^k$ and $\mathbf{u} \leftarrow \mathbb{Z}_p^\ell$.

Definition 3.3.2 ($\mathcal{D}_{\ell,k}$-Matrix Diffie-Hellman Assumption $\mathcal{D}_{\ell,k} - \mathsf{MDDH}$)**.** *Let $\mathcal{D}_{\ell,k}$ be a matrix distribution. We say that the $\mathcal{D}_{\ell,k}$-Matrix Diffie-Hellman ($\mathcal{D}_{\ell,k} - \mathsf{MDDH}$) Assumption holds relative to $\mathcal{G}$ if the distributions of $(A||A\mathbf{r})\bar{g}$ and $(A||\mathbf{u})\bar{g}$ are computationally indistinguishable, where $(\bar{\mathbb{G}}, p, \bar{g}) \leftarrow \mathcal{G}$, $A \leftarrow \mathcal{D}_{\ell,k}$, $\mathbf{w} \leftarrow \mathbb{Z}_p^k$ and $\mathbf{u} \leftarrow \mathbb{Z}_p^\ell$.*

Due to its linearity properties, the $\mathcal{D}_{\ell,k} - \mathsf{MDDH}$ assumption does not hold in $(k+1)$-linear groups, assuming that k is constant, i.e. it does not depend on the security parameter.

Lemma 3.3.3. *Let $\mathcal{D}_{\ell,k}$ be any matrix distribution. Then the $\mathcal{D}_{\ell,k}$-Matrix Diffie-Hellman Assumption is false in $(k+1)$-linear groups.*

Proof. In a $(k+1)$-linear group, the implicit representation of any $r \times r$ determinant for $r \leq k+1$ can be efficiently computed by using the r-linear map given by the Leibnitz formula:

$$\det(M) = \sum_{\sigma \in S_r} \operatorname{sgn}(\sigma) \prod_{i=1}^{r} m_{i,\sigma_i}$$

Using the $(k+1)$-linear map, $\det(M)e(\bar{g}, \overset{(k)}{\ldots}, \bar{g})$ can be computed in the target group. Then, given $B\bar{g} := (A||\mathbf{z})\bar{g}$, consider the submatrix A_0 formed by the first k rows of A and the vector $\mathbf{z}_0$ formed by the first k elements of $\mathbf{z}$.

If $\det(A_0) \neq 0$, then define C as the submatrix formed by the first $k+1$ rows of B. If $\mathbf{z}$ is random then $\det(C) \neq 0$ with overwhelming probability, while if $\mathbf{z} = A\mathbf{w}$ for some vector $\mathbf{w}$ then $\det(C) = 0$. Therefore the $\mathcal{D}_{\ell,k}$-Matrix Diffie-Hellman Assumption is false in this case.

Otherwise $\det(A_0) = 0$. Then $\operatorname{rank}(A_0||\mathbf{z}_0) = \operatorname{rank}(A_0)$ when $\mathbf{z} = A\mathbf{w}$, while $\operatorname{rank}(A_0||\mathbf{z}_0) = \operatorname{rank}(A_0) + 1$ with overwhelming probability if $\mathbf{z}$ is random. To compute the rank of both matrices the following efficient randomized algorithm can be used. Take random invertible matrices $L, R \in \mathbb{Z}_p^{k\times k}$. Then set $A_0'\bar{g} = LA_0R\bar{g}$ and $\mathbf{z}_0'\bar{g} = L\mathbf{z}_0\bar{g}$, which is just a randomized instance of the same problem. Now if $\operatorname{rank}(A_0') = r$ then with overwhelming probability its principal $r \times r$ minor is nonzero. Therefore, we can estimate $r = \operatorname{rank}(A_0')$ as the size of the largest nonzero principal minor (with negligible error probability). Finally, if the determinant of the submatrix of $A_0'||\mathbf{z}_0'$ formed by the first $r+1$ rows and the first r and the last column is nonzero we conclude that $\mathbf{z}$ is random. □

3.3.1 Examples of $\mathcal{D}_{\ell,k} - \mathsf{MDDH}$

Let $\mathcal{D}_{\ell,k}$ be a matrix distribution and $A \leftarrow \mathcal{D}_{\ell,k}$. We define the representation size $\mathsf{RE}_{\bar{\mathbb{G}}}(\mathcal{D}_{\ell,k})$ of a given polynomial-induced matrix distribution $\mathcal{D}_{\ell,k}$ as the minimal number of group elements it takes to represent $A\bar{g}$ for any $A \in \mathcal{D}_{\ell,k}$. By Lemma 3.3.3 we obtain a family of strictly weaker assumptions. Our goal is to obtain such a family of assumptions with small (possibly minimal) representation.

Example 1. *Let $\mathcal{U}_{\ell,k}$ be the uniform distribution over $\mathbb{Z}_p^{\ell\times k}$.*

The next lemma says that $\mathcal{U}_{\ell,k} - \mathsf{MDDH}$ is the weakest possible assumption among all $\mathcal{D}_{\ell,k}$-Matrix Diffie-Hellman Assumptions. However, $\mathcal{U}_{\ell,k}$ has a poor representation, i.e., $\mathsf{RE}_{\bar{\mathbb{G}}}(\mathcal{U}_{\ell,k}) = \ell k$.

Lemma 3.3.4. *Let $\mathcal{D}_{\ell,k}$ be any matrix distribution. Then $\mathcal{D}_{\ell,k} \implies \mathcal{U}_{\ell,k}$.*

Proof. Given an instance $(A\bar{g}, A\mathbf{w}\bar{g})$ of $\mathcal{D}_{\ell,k}$, if $L \in \mathbb{Z}_p^{\ell\times\ell}$ and $R \in \mathbb{Z}_p^{k\times k}$ are two random invertible matrices, it is possible to get a properly distributed instance of the $\mathcal{U}_{\ell,k}$-matrix DH problem as $(LAR\bar{g}, LA\mathbf{w}\bar{g})$. Indeed, LAR has a distribution statistically close to the uniform distribution[2] in $\mathbb{Z}_p^{k\times\ell}$, while $LA\mathbf{w} = LAR\mathbf{v}$ for $\mathbf{v} = R^{-1}\mathbf{w}$. Clearly, $\mathbf{v}$ has the uniform distribution in $\mathbb{Z}_p^k$. □

Example 2 (k-Linear Assumption / $k - \mathsf{Lin}$)**.** *We define the distribution $\mathcal{L}_k$ as follows*

$$A = \begin{pmatrix} a_1 & 0 & \cdots & 0 & 0 \\ 0 & a_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & a_k \\ 1 & 1 & \cdots & 1 & 1 \end{pmatrix}$$

[2] If A has full-rank (that happens with overwhelming probability) then LAR is uniformly distributed in the set of full-rank matrices in $\mathbb{Z}_p^{\ell\times k}$, which implies that it is close to uniform in $\mathbb{Z}_p^{\ell\times k}$.

where $a_i \leftarrow \mathbb{Z}_p^*$. Note that the distribution $(A, A\mathbf{w})$ can be compactly written as $(a_1, \ldots, a_k, a_1w_1, \ldots, a_kw_k, w_1 + \cdots + w_k) = (a_1, \ldots, a_k, b_1, \ldots, b_k, \frac{b_1}{a_1} + \cdots + \frac{b_k}{a_k}$ with $a_i \leftarrow \mathbb{Z}_p^*, b_i, w_i \leftarrow \mathbb{Z}_p$. Hence the $\mathcal{L}_k$-Matrix Diffie-Hellman Assumption is an equivalent description of the k-linear Assumption [BBS04, HK07, Sha07], with $\mathsf{RE}_{\tilde{\mathbb{G}}}(\mathcal{L}_k) = k$.

Example 3 (k-Cascade Assumption / $k - \mathsf{Casc}$)**.** *We define the distribution $\mathcal{C}_k$ as follows*

$$A = \begin{pmatrix} a_1 & 0 & \cdots & 0 & 0 \\ 1 & a_2 & \cdots & 0 & 0 \\ 0 & 1 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & a_k \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}$$

where $a_i \leftarrow \mathbb{Z}_p^*$. Note that $(A, A\mathbf{w})$ can be compactly written as $(a_1, \ldots, a_k, a_1w_1, w_1 + a_2w_2, \ldots, w_{k-1} + a_kw_k, w_k) = (a_1, \ldots, a_k, b_1, \ldots, b_k, \frac{b_k}{a_k} - \frac{b_{k-1}}{a_{k-1}a_k} + \frac{b_{k-2}}{a_{k-2}a_{k-1}a_k} - \cdots \pm \frac{b_1}{a_1 \cdot \cdots \cdot a_k}$. We have $\mathsf{RE}_{\tilde{\mathbb{G}}}(\mathcal{C}_k) = k$.

Matrix A bears resemblance to a cascade which explains the assumption's name. Indeed, in order to compute the right lower entry w_k of matrix $(A, A\mathbf{w})$ from the remaining entries, one has to "descend" the cascade to compute all the other entries w_i $(1 \leq i \leq k-1)$ one after the other.

A more compact version of $\mathcal{C}_k$ is obtained by setting all $a_i := a$.

Example 4 (Symmetric k-Cascade Assumption / $k - \mathsf{SCasc}$)**.** *We define the distribution $\mathcal{SC}_k$ as $\mathcal{C}_k$ but now $a_i = a$, where $a \leftarrow \mathbb{Z}_p^*$. Then $(A, A\mathbf{w})$ can be compactly written as $(a, aw_1, w_1 + aw_2, \ldots, w_{k-1} + aw_k, w_k) = (a, b_1, \ldots, b_k, \frac{b_k}{a} - \frac{b_{k-1}}{a^2} + \frac{b_{k-2}}{a^3} - \cdots \pm \frac{b_1}{a^k})$. We have $\mathsf{RE}_{\tilde{\mathbb{G}}}(\mathcal{SC}_k) = 1$.*

Observe that the same trick cannot be applied to the k-Linear assumption $k - \mathsf{Lin}$, as the resulting Symmetric k-Linear assumption does not hold in k-linear groups. However, if we set $a_i := a + i - 1$, we obtain another matrix distribution with compact representation.

Example 5 (Incremental k-Linear Assumption)**.** *We define the distribution $\mathcal{IL}_k$ as $\mathcal{L}_k$ with $a_i = a + i - 1$, for $a \leftarrow \mathbb{Z}_p^*$. $(A, A\mathbf{w})$ can be compactly written as $(a, aw_1, (a+1)w_2, \ldots, (a+k-1)w_k, w_1 + \cdots + w_k) = (a_1, b_1, \ldots, b_k, \frac{b_1}{a} + \frac{b_2}{a+1} + \cdots + \frac{b_k}{a+k-1})$. We also have $\mathsf{RE}_{\tilde{\mathbb{G}}}(\mathcal{IL}_k) = 1$..*

The following theorem is proven in [EHK$^+$17], which shows the relation between these new assumptions.

Theorem 3.3.5. *For any $k \geq 2$, the following holds:*

$$k - \mathsf{SCasc} \Longleftrightarrow k - \mathsf{ILin};$$
$$k - \mathsf{SCasc} \implies k - \mathsf{Casc}; \quad k - \mathsf{ILin} \implies k - \mathsf{Lin};$$
$$k - \mathsf{Casc} \implies (k+1)\mathsf{Casc}; \quad k - \mathsf{SCasc} \implies (k+1)\mathsf{SCasc}$$

3.4 Instantiating Groth-Sahai proofs using the MDDH-assumptions

As we explained in Section 2.8, in [GS12] Groth and Sahai give a framework for creating NIWI and NIZK proofs for the language of satisfiability of a set of quadratic equations over a bilinear group.

In order to instantiate the framework, we need to show how the Setup, the GenCRS and the $\mathsf{SimGenCRS}$ algorithms work, this is, how the common reference string is created in the soundness setting and in the simulation setting. In addition, we must show that their outputs are indistinguishable and that they satisfy the properties required by the framework. Indeed, the other algorithms ($\mathsf{Prove}, \mathsf{SimProve}, \mathsf{VerifyProof}$) are completely described by the Groth-Sahai framework once the common reference string is known.

Setup : $gk := (p, \bar{\mathbb{G}}, \mathbb{T}, e, \bar{g}) \leftarrow \mathcal{SPG}_{\mathcal{D}_{\ell,k}}(1^\lambda)$[3]

GenCRS : On input gk, return $\mathsf{crs} := (\bar{\mathbf{u}}_1, \ldots, \bar{\mathbf{u}}_{k+1})$, constructed as follows. First, the algorithm picks $A \leftarrow \mathcal{D}_{\ell,k}$, $\mathbf{w} \leftarrow \mathbb{Z}_p^k$ and a vector $\mathbf{z} \in \mathbb{Z}_p^\ell, \mathbf{z} \notin \mathsf{Im}(A)$, this is, not in the span of the columns of A. Then, $(\bar{\mathbf{u}}_1, \ldots, \bar{\mathbf{u}}_{k+1})$ are defined as the columns of $(A||A\mathbf{w})\bar{g}$. For notational convenience, we define $\bar{U} = (\bar{\mathbf{u}}_1, \ldots, \bar{\mathbf{u}}_{k+1}) = (A||A\mathbf{w})\bar{g}$ and $\bar{U}' = (\bar{\mathbf{u}}_1, \ldots, \bar{\mathbf{u}}_k) = A\bar{g}$.

The modules $\mathfrak{B}_1, \mathfrak{B}_2, \mathfrak{B}_T$ are implicitly defined as $\bar{\mathbb{G}}^\ell, \bar{\mathbb{G}}^\ell, \mathbb{T}^{\ell\times\ell}$ respectively, and are used for all equation types. Let $\boldsymbol{\xi} \in \mathbb{Z}_p^\ell$ be an arbitrary vector such that $\boldsymbol{\xi}^\top A = \mathbf{0}$ and $\boldsymbol{\xi}^\top \mathbf{z} = 1$. The maps are defined as follows:

- Two different bilinear maps F, F_{sym} are defined. The map $F : \bar{\mathbb{G}}^\ell \times \bar{\mathbb{G}}^\ell \to \mathbb{T}^{\ell\times\ell}$ is defined as

$$F : \left(\begin{pmatrix} \bar{x}_1 \\ \vdots \\ \bar{x}_\ell \end{pmatrix}, \begin{pmatrix} \bar{y}_1 \\ \vdots \\ \bar{y}_\ell \end{pmatrix} \right) \mapsto \begin{pmatrix} e(\bar{x}_1, \bar{y}_1) & \cdots & e(\bar{x}_1, \bar{y}_\ell) \\ \vdots & \ddots & \vdots \\ e(\bar{x}_\ell, \bar{y}_1) & \cdots & e(\bar{x}_\ell, \bar{y}_\ell) \end{pmatrix},$$

  whereas the symmetric map F_{sym} is defined by

$$F_{sym}(\bar{\mathbf{x}}, \bar{\mathbf{y}}) := \frac{1}{2}\tilde{F}(\bar{\mathbf{x}}, \bar{\mathbf{y}}) + \frac{1}{2}\tilde{F}(\bar{\mathbf{y}}, \bar{\mathbf{x}})$$

- The maps $\iota : \bar{\mathbb{G}} \to \bar{\mathbb{G}}^\ell$ and $\mathfrak{p} : \bar{\mathbb{G}}^\ell \to \bar{\mathbb{G}}$ are defined as:

$$\iota(\bar{x}) := \mathbf{z}\bar{x}, \qquad \mathfrak{p}(\bar{\mathbf{c}}) = \boldsymbol{\xi}^\top \bar{\mathbf{c}},$$

- The maps $\iota' : \mathbb{Z}_p \to \bar{\mathbb{G}}^\ell$ and $\mathfrak{p}' : \bar{\mathbb{G}}^\ell \to \mathbb{Z}_p$ are defined as:

$$\iota'(x) = x(\bar{\mathbf{u}}_{k+1} + \mathbf{z}\bar{g}), \qquad \mathfrak{p}'(\bar{\mathbf{c}}) = \log_{\bar{g}}(\boldsymbol{\xi}^\top \bar{\mathbf{c}}),$$

  where $\log_{\bar{g}}$ is the discrete logarithm with respect to $\bar{g}$.

[3] We write $\mathcal{SPG}_{\mathcal{D}_{\ell,k}}$ to say that the $\mathcal{D}_{\ell,k}$ assumption holds relative to $\mathcal{SPG}$, where the definition of $\mathcal{D}_{\ell,k}$ holding relative to $\mathcal{SPG}$ is the natural extension of Definition 3.3.2.

In the following, let $H^{r,s,m,n} = (h_{ij}) \in \mathbb{Z}_p^{m\times n}$ denote the matrix such that $h_{rs} = -1$, $h_{sr} = 1$ and $h_{ij} = 0$ for $(i,j) \notin \{(r,s),(s,r)\}$. The matrices $\{H_i\}$ and the maps $\iota_T, \mathfrak{p}_T$ are defined as follows for each equation type:

Pairing product equation: Commitment keys $\bar{U}$ and the inclusion map ι are used to commit to elements in $\bar{\mathbb{G}}$. The maps $\iota_T, \mathfrak{p}_T$ are the same for both F and F_{sym}:

$$\iota_T(z_{\mathbb{T}}) = \mathbf{z}\cdot\mathbf{z}^\top z_{\mathbb{T}} \in \mathbb{T}^{\ell\times\ell} \qquad\qquad \mathfrak{p}_T(Z_{\mathbb{T}}) = \boldsymbol{\xi}^\top Z_{\mathbb{T}}\boldsymbol{\xi}_T,$$

where $z_{\mathbb{T}} \in \mathbb{T}$, $Z_{\mathbb{T}} \in \mathbb{T}^{\ell\times\ell}$. The equation $\bar{U}H \bullet \bar{U} = 0_{\mathbb{T}}$ admits no non-trivial solution, while all the solutions to $\bar{U}H \bullet_{sym} \bar{U} = 0_{\mathbb{T}}$ are generated by $\left\{H^{r,s,k+1,k+1}\right\}_{1\le r<s\le k+1}$.

Multiscalar multiplication equations: Commitment keys $\bar{U}$ and $\bar{U}'$ and inclusion maps ι and ι' are used to commit to elements in $\bar{\mathbb{G}}$ and $\mathbb{Z}_p$ respectively. Both for $\tilde{F} = F$ and $\tilde{F} = F_{sym}$, $\tilde{\iota}_T$, $\tilde{\mathfrak{p}}_T$ are defined as:

$$\tilde{\iota}_T(\bar{z}) := \tilde{F}(\iota'(1), \iota(\bar{z})), \qquad \tilde{\mathfrak{p}}_T(Z_{\mathbb{T}}) := \boldsymbol{\xi}^\top \bar{Z}\boldsymbol{\xi},$$

where $\bar{Z} \in \bar{\mathbb{G}}^{\ell\times\ell}$ is such that $e(\bar{Z}_{ij}, \bar{g}) = (Z_{\mathbb{T}})_{ij}$ for all entries of $\bar{Z}$. The equation $\bar{U}'H \bullet \bar{U} = \bar{0}$ admits no solution, while all the solutions to $\bar{U}'H \bullet_{sym} \bar{U} = \bar{0}$ are generated by $\left\{H^{r,s,k,k+1}\right\}_{1\le r<s\le k}$.

Quadratic equations: Commitment keys $\bar{U}'$ and the inclusion map ι' are used to commit to elements in $\mathbb{Z}_p$. Both for $\tilde{F} = F$ and $\tilde{F} = F_{sym}$, $\tilde{\iota}'_T$, $\tilde{\mathfrak{p}}'_T$ are defined as:

$$\tilde{\iota}'_T(z) := \tilde{F}(\iota'(1), \iota'(z)), \qquad \tilde{\mathfrak{p}}'_T(Z_{\mathbb{T}}) := \boldsymbol{\xi}^\top Z\boldsymbol{\xi},$$

where $Z \in \mathbb{Z}_p^{\ell\times\ell}$ is the matrix which entries are the discrete logarithms of the entries of $Z_{\mathbb{T}}$ with respect to $e(\bar{g}, \bar{g})$. The equation $\bar{U}'H \bullet \bar{U}' = \bar{0}$ admits no solution, while all the solutions to $\bar{U}'H \bullet_{sym} \bar{U}' = \bar{0}$ are generated by $\left\{H^{r,s,k,k}\right\}_{1\le r<s\le k}$.

SimGenCRS : On input gk, return crs $:= (\bar{\mathbf{u}}_1, \ldots, \bar{\mathbf{u}}_{k+1})$ and the trapdoor key $tk = \mathbf{s}_{tk} := \mathbf{w}$, constructed as follows. First, the algorithm picks $A \leftarrow \mathcal{D}_{\ell,k}$, $\mathbf{w} \leftarrow \mathbb{Z}_p^k$ and a vector $\mathbf{z} \in \mathbb{Z}_p^\ell$, $\mathbf{z} \notin \mathsf{Im}(A)$, this is, not in the span of the columns of A. Then, $(\bar{\mathbf{u}}_1, \ldots, \bar{\mathbf{u}}_{k+1})$ are defined as the columns of $(A||A\mathbf{w} - \mathbf{z})\bar{g}$. The required maps and matrices are implicitly defined exactly as in the GenCRS algorithm.

The Prove and VerifyProof algorithms are the same as those for the DLIN instantiation of [GS12], given in Section 2.8.6, and the same optimizations can be used.

Theorem 3.4.1. *The protocol given above is a NIWI proof with perfect completeness, perfect soundness and composable witness-indistinguishability for satisfiability of a set of equations over a symmetric bilinear* *group $(p, \bar{\mathbb{G}}, \mathbb{T}, e, \bar{g})$ if the $\mathcal{D}_{\ell,k}$-Matrix Diffie-Hellman Assumption holds relative to $\mathcal{SPG}_{\mathcal{D}_{\ell,k}}$. The protocol is zero-knowledge if for all pairing product equations, the term on the right-hand side of the equation, $t_{\mathbb{T}}$, is of the form $t_{\mathbb{T}} = \prod_{i=1}^{n} e(\bar{p}_i, \bar{q}_i)$ for public $\bar{p}_i, \bar{q}_i$ and the equations are rewritten so that they can be simulated as explained in Section 2.8.4.*

Proof. Since this is an instantiation of the Groth-Sahai proof framework, we just need to prove that all the requirements of the framework are satisfied. In particular, we need to show that:

1. The requirements for the crs generated by the GenCRS algorithm are satisfied:

   - For each equation type, the corresponding maps $\iota, \mathfrak{p}, \iota_T, \mathfrak{p}_T, F$ and f commute, this is:

   $$\forall x \in \mathfrak{A}_1, \forall y \in \mathfrak{A}_2 : F(\iota(x), \iota(y)) = \iota_T(f(x, y)),$$
   $$\forall x \in \mathfrak{B}_1, \forall y \in \mathfrak{B}_2 : f(\mathfrak{p}(x), \mathfrak{p}(y)) = \mathfrak{p}_T(F(x, y)).$$

   - For each equation type, the maps ι,ι_T and F are efficiently computable.
   - For each equation type, $\mathfrak{p}_1(\mathbf{u}) = \mathbf{0}$, $\mathfrak{p}_2(\mathbf{v}) = \mathbf{0}$ and the maps $\mathfrak{p}_1 \circ \iota_1$, $\mathfrak{p}_2 \circ \iota_2$ and $\mathfrak{p}_T \circ \iota_T$ are the identity map in their respective domains.

   It is easy to see that all these requirements are satisfied by construction.

2. The requirements for the crs generated by the SimGenCRS algorithm are satisfied:

   - The maps $\iota_1, \mathfrak{p}_1, \iota_2, \mathfrak{p}_2 \iota_T, \mathfrak{p}_T, F$ and f must commute, this is:

   $$\forall x \in \mathfrak{A}_1, \forall y \in \mathfrak{A}_2 : F(\iota_1(x), \iota_2(y)) = \iota_T(f(x, y)),$$
   $$\forall x \in \mathfrak{B}_1, \forall y \in \mathfrak{B}_2 : f(\mathfrak{p}_1(x), \mathfrak{p}_2(y)) = \mathfrak{p}_T(F(x, y)).$$

   - The maps ι_1,ι_2, ι_T and F must be efficiently computable.
   - $\iota_1(\mathfrak{A}_1) \subseteq \langle u_1, \ldots, u_{\hat{m}} \rangle$ and $\iota_1(\mathfrak{A}_2) \subseteq \langle v_1, \ldots, v_{\hat{n}} \rangle$.
   - $H_1, \ldots, H_\eta$ generate the $\mathcal{R}$-module of all matrices $H \in \mathcal{R}^{\hat{m} \times \hat{n}}$ such that $\mathbf{u}H \bullet \mathbf{v} = 0$.

   The first three requirements are satisfied by construction. Regarding the fourth one, it follows from the observation that if $u_1, \ldots, u_k$ are linearly independent, then $F(u_i, u_j)$ are also linearly independent.

3. The crs distributions generated by the GenCRS and the SimGenCRS algorithms are computationally indistinguishable. This is a direct consequence of the assumption that the $\mathcal{D}_{\ell,k}$-Matrix Diffie-Hellman Assumption holds relative to $\mathcal{SPG}_{\mathcal{D}_{\ell,k}}$.

4. The trapdoor key $tk = \mathbf{s}_{tk}$ is such that $\iota'(1) = \iota'(0) + \mathbf{u}'\mathbf{s}_{tk}$, which is satisfied by construction.

□

Assumption: $\mathcal{D}_{\ell,k}$	$\bar{\mathbb{G}}$	$\mathbb{Z}_p$
Variables $x \in \mathbb{Z}_p$, $\bar{x} \in \bar{\mathbb{G}}$	ℓ	0

Figure 3.1: Cost of committing to each variable type.

Assumption: $\mathcal{D}_{\ell,k}$	$\bar{\mathbb{G}}$	$\mathbb{Z}_p$
Pairing product equations	$\ell(k+1)$	0
- Linear equation: $\bar{\mathbf{a}} \cdot \bar{\mathbf{y}} = t_{\mathbb{T}}$	$k+1$	0
Multiscalar multiplication equations:	$\ell(k+1)$	0
- Linear equation: $\mathbf{a} \cdot \bar{\mathbf{y}} = \bar{t}$	0	$k+1$
- Linear equation: $\mathbf{x} \cdot \bar{\mathbf{b}} = \bar{t}$	k	0
Quadratic equations in $\mathbb{Z}_p$	ℓk	0
- Linear equation: $\mathbf{x} \cdot \mathbf{b} = t$	0	k

Figure 3.2: Size of an NIZK proof for each type of equation.

Size. By applying the optimizations used in Section 2.8.6, it follows that the cost of each proof can be derived from the figures Fig. 3.1 and Fig. 3.2.

We emphasize that for $\mathcal{D}_{\ell,k} = \mathcal{L}_2$ and $\mathbf{z} = (0,0,1)^\top$ and for $\mathcal{D}_{\ell,k} = \mathcal{L}_1$ and $\mathbf{z} = (0,1)^\top$ (in the natural extension to asymmetric bilinear groups), we recover the $2-\mathsf{Lin}$ and the SXDH instantiations of [GS12]. While the size of the proofs depends only on ℓ and k, both the size of the crs and the cost of verification increase with $\mathsf{RE}_{\bar{\mathbb{G}}}(\mathcal{D}_{\ell,k})$. In particular, in terms of efficiency, the $\mathcal{SC}_2$ Assumption is preferable to the $2-\mathsf{Lin}$ Assumption but the main reason to consider more instantiations of Groth-Sahai proofs is to obtain more efficient proofs for a large class of languages in Section 3.5.

3.5 More efficient proofs for some crs dependent languages

Let $\bar{U}$ be the commitment key defined in last section as part of a $\mathcal{D}_{\ell,k}$-MDDH instantiation, for some $A \leftarrow \mathcal{D}_{\ell,k}$. In this section, we show how to shorter proofs of some languages related to A. The common idea of all the improvements is to exploit the special structure of the homomorphic commitments used in Groth Sahai proofs. Our proofs implicitly use the Groth-Sahai framework, although we have preferred to give the proofs without using the Groth-Sahai notation. In addition, in this section we use simplified notation for a bilinear pairing, writing $e(\bar{x}, \bar{y}) = \bar{x}\bar{u}$ and

$$\bar{\mathbf{x}}\bar{\mathbf{y}}^\top = \begin{pmatrix} e(\bar{x}_1, \bar{y}_1) & \cdots & e(\bar{x}_1, \bar{y}_\ell) \\ \vdots & \ddots & \vdots \\ e(\bar{x}_\ell, \bar{y}_1) & \cdots & e(\bar{x}_\ell, \bar{y}_\ell) \end{pmatrix}$$

3.5.1 More efficient subgroup membership proofs

We first show how to obtain shorter membership proofs in the language $\mathcal{L}_{A,gk} := \{A\mathbf{r}\bar{g}, \mathbf{r} \in \mathbb{Z}_p^k\} \subset \bar{\mathbb{G}}^\ell$.

Intuition

The idea behind our improvement is to exploit the special algebraic structure of commitments in Groth-Sahai proofs, namely the observation that if $\bar{\mathbf{\Phi}} = A\mathbf{r}\bar{g} \in \mathcal{L}_{A,gk}$ then $\bar{\mathbf{\Phi}}$ can be seen as a commitment of the scalar 0 using randomness $\mathbf{r}$. Therefore, to prove that $\bar{\mathbf{\Phi}} \in \mathcal{L}_{A,gk}$, we proceed as if we were giving a Groth-Sahai proof of satisfiability of the equation $x = 0$ where the randomness used for the commitment to x is $\mathbf{r}$. In particular, no commitments have to be given in the proof, which results in shorter proofs. To prove zero-knowledge we rewrite the equation $x = 0$ as $x \cdot \delta = 0$. The real proof is just a standard Groth-Sahai proof with the commitment to $\delta = 1$ being $\iota'(1)$ a commitment to the scalar 1 using randomness $\mathbf{0}$, while in the simulated proof the trapdoor allows to open $\iota'(1)$ as a commitment of 0, so we can proceed as if the equation was the trivial one $x \cdot 0 = 0$, for which it is easy to give a proof of satisfiability.

The construction

Define $\mathcal{H} := \{H \in \mathbb{Z}_p^{k\times k} : H + H^\top = 0\}$. Following the intuition given above, the actual construction looks as follows:

Setup : the setup samples and outputs a group $(p, \bar{\mathbb{G}}, \mathbb{T}, e, \bar{g}) \leftarrow \mathcal{SPG}(1^\lambda)$.

GenCRS : The commitment keys $\bar{U} = (\bar{\mathbf{u}}_1, \ldots, \bar{\mathbf{u}}_{k+1})$ are defined as $(A||A\mathbf{w} + \mathbf{z})\bar{g}$, where $A \leftarrow \mathcal{D}_{\ell,k}$, $\mathbf{w} \leftarrow \mathbb{Z}_p^k$, and $\mathbf{z} \in \mathbb{Z}_p^\ell$, $\mathbf{z} \notin \mathsf{Im}(A)$. For convenience, we define $\bar{U}' = (\bar{\mathbf{u}}_1, \ldots, \bar{\mathbf{u}}_k) = A\bar{g}$. The common reference string is $\mathsf{crs} := (\bar{U}, \mathbf{z})$.

$\mathsf{SimGenCRS}$: The commitment key $\bar{U} = (\bar{\mathbf{u}}_1, \ldots, \bar{\mathbf{u}}_{k+1})$ are defined as $(A||A\mathbf{w})\bar{g}$, where $A \leftarrow \mathcal{D}_{\ell,k}$, $\mathbf{w} \leftarrow \mathbb{Z}_p^k$, and $\mathbf{z} \in \mathbb{Z}_p^\ell$, $\mathbf{z} \notin \mathsf{Im}(A)$. The common reference string is $\mathsf{crs} := (\bar{U}, \mathbf{z})$. The simulation trapdoor τ is the vector $\mathbf{w} \in \mathbb{Z}_p^k$.

Prove : On input crs, a vector $\bar{\mathbf{\Phi}} = A\mathbf{r}\bar{g} \in \mathcal{L}_{A,gk}$ and the witness $\mathbf{r} \in \mathbb{Z}_p^k$, the prover chooses a matrix $H \leftarrow \mathcal{H}$ and computes

$$\bar{\Pi} = \bar{\mathbf{u}}_{k+1}\mathbf{r}^\top + \bar{U}'H.$$

$\mathsf{VerifyProof}$: On input crs, $\bar{\mathbf{\Phi}}, \bar{\Pi}$, the verifier checks if $\bar{\mathbf{\Phi}}\bar{\mathbf{u}}_{k+1}^\top + \bar{\mathbf{u}}_{k+1}\bar{\mathbf{\Phi}}^\top = \bar{\Pi}\bar{U}'^\top + \bar{U}'\bar{\Pi}^\top$.

$\mathsf{SimProof}$. On input crs, $\bar{\mathbf{\Phi}}, \tau$ the simulator picks a matrix $H_{\mathsf{sim}} \leftarrow \mathcal{H}$ and computes

$$\bar{\Pi}_{\mathsf{sim}} = \bar{\mathbf{\Phi}}\mathbf{w}^\top + \bar{U}'H_{\mathsf{sim}}.$$

Theorem 3.5.1. *The proof system given above is a Non-Interactive Zero-Knowledge Proof for the language $\mathcal{L}_{A,gk}$, with perfect completeness, perfect soundness and composable zero-knowledge of $k\ell$ group elements based on the $\mathcal{D}_{\ell,k}$-*MDDH *Assumption.*

Proof. First, it is clear that under the $\mathcal{D}_{\ell,k}$-MDDH Assumption, the distribution of crs generated by the GenCRS algorithm and the distribution of crs generated by the $\mathsf{SimGenCRS}$ algorithm are computationally indistinguishable. For the sake of simplicity, we talk about the soundness setting and the WI setting for referring to when the crs is created by the GenCRS algorithm or the $\mathsf{SimGenCRS}$ algorithm respectively.

Completeness. To see completeness, we see that a real proof satisfies the verification equation. Indeed, in the soundness setting, the left term of the verification equation is:

$$\begin{aligned}\bar{\mathbf{\Phi}}\bar{\mathbf{u}}_{k+1}^\top + \bar{\mathbf{u}}_{k+1}\bar{\mathbf{\Phi}}^\top &= (A\mathbf{r}\bar{g})((A\mathbf{w}+\mathbf{z})\bar{g})^\top + ((A\mathbf{w}+\mathbf{z})\bar{g})(A\mathbf{r}\bar{g})^\top \\ &= (A(\mathbf{r}\mathbf{w}^\top + \mathbf{w}\mathbf{r}^\top)A^\top + A\mathbf{r}\mathbf{z}^\top + \mathbf{z}\mathbf{r}^\top A^\top)\bar{g}\bar{g},\end{aligned}$$

while the right term in the real proof is:

$$\begin{aligned}\bar{\Pi}\bar{U}'^\top + \bar{U}'\bar{\Pi}^\top &= (A(\mathbf{w}\mathbf{r}^\top + \mathbf{w}\mathbf{r}^\top)A^\top + A(H+H^\top)A^\top + A\mathbf{r}\mathbf{z}^\top + \mathbf{z}\mathbf{r}^\top A^\top)\bar{g}\bar{g} \\ &= (A(\mathbf{r}\mathbf{w}^\top + \mathbf{w}\mathbf{r}^\top)A^\top + A\mathbf{r}\mathbf{z}^\top + \mathbf{z}\mathbf{r}^\top A^\top)\bar{g}\bar{g}\end{aligned}$$

Soundness. Let $\boldsymbol{\xi} \in \mathbb{Z}_p^\ell$ be any vector such that $\boldsymbol{\xi}^\top A = \mathbf{0}, \boldsymbol{\xi}^\top \mathbf{z} = 1$. This implies that $\boldsymbol{\xi}^\top \bar{U}' = \bar{0}$ and, in the soundness setting, $\boldsymbol{\xi}^\top \bar{\mathbf{u}}_{k+1} = \bar{g}$. Therefore, if $\bar{\Pi}$ is any proof that satisfies the verification equation, multiplying on the left by $\boldsymbol{\xi}^\top$ and the right by $\boldsymbol{\xi}$,

$$\boldsymbol{\xi}^\top(\bar{\Phi}\bar{\mathbf{u}}_{k+1}^\top + \bar{\mathbf{u}}_{k+1}\bar{\Phi}^\top)\boldsymbol{\xi} = \boldsymbol{\xi}^\top(\bar{\Pi}\bar{U}'^\top + \bar{U}\bar{\Pi}^\top)\boldsymbol{\xi},$$

we obtain

$$\boldsymbol{\xi}^\top\bar{\mathbf{\Phi}}\bar{g} + \bar{g}\bar{\mathbf{\Phi}}^\top\boldsymbol{\xi} = 0_{\mathbb{T}}$$

Since $\boldsymbol{\xi}^\top\bar{\mathbf{\Phi}} + \bar{\mathbf{\Phi}}^\top\boldsymbol{\xi} = 2\boldsymbol{\xi}^\top\bar{\mathbf{\Phi}}$, from this last equation it follows that $\boldsymbol{\xi}^\top\bar{\mathbf{\Phi}}\bar{g} = 0_{\mathbb{T}}$. This holds for any vector $\boldsymbol{\xi}$ such that $\boldsymbol{\xi}^\top A = 0$ and $\boldsymbol{\xi}^\top\mathbf{z} = 1$, which implies that $\bar{\mathbf{\Phi}} \in \mathcal{L}_{A,gk}$ which proves perfect soundness.

Composable Zero-Knowledge. We will now see that, in the witness indistinguishability setting, both a real proof and a simulated proof have the same distribution when $\bar{\mathbf{\Phi}} \in \mathcal{L}_{A,gk}$. We first note that they both satisfy the verification equation. Indeed, the left term of the verification equation in the WI setting is

$$\bar{\mathbf{\Phi}}\bar{\mathbf{u}}_{k+1}^\top + \bar{\mathbf{u}}_{k+1}\bar{\mathbf{\Phi}}^\top = A(\mathbf{r}\mathbf{w}^\top + \mathbf{w}\mathbf{r}^\top)A^\top\bar{g}\bar{g},$$

which is obviously equal to the right term of the verification equation for the real proof. On the other hand, if $\bar{\mathbf{\Phi}} \in \mathcal{L}_{A,gk}$, the right term of the verification equation for a simulated proof is:

$$\begin{aligned}\bar{\Pi}_{\mathsf{sim}}\bar{U}'^\top + \bar{U}'\bar{\Pi}_{\mathsf{sim}}^\top &= A(\mathbf{r}\mathbf{w}^\top + \mathbf{w}\mathbf{r}^\top)A^\top\bar{g}\bar{g} + A(H' + (H')^\top)A^\top\bar{g}\bar{g} \\ &= A(\mathbf{r}\mathbf{w}^\top + \mathbf{w}\mathbf{r}^\top)A^\top\bar{g}\bar{g},\end{aligned}$$

for some $H_{\mathsf{sim}} \in \mathcal{H}$.

We now argue that an honestly generated proof $\bar{\Pi}$ and a simulated proof $\bar{\Pi}_{\mathsf{sim}}$ have the same distribution. By construction, there exist some matrices Θ and Θ' such that $\bar{\Pi} = \bar{U}'\Theta$ and $\bar{\Pi}_{\mathsf{sim}} = \bar{U}'\Theta'$. Now, if $\bar{\Pi}_1 = \bar{U}'\Theta_1$ and $\bar{\Pi}_2 = \bar{U}'\Theta_2$ are two proofs, real or simulated, which satisfy the verification equation, then necessarily $(\bar{\Pi}_1 - \bar{\Pi}_2)\bar{U}'^{\top} + \bar{U}'(\bar{\Pi}_1 - \bar{\Pi}_2)_T = A((\Theta_1 - \Theta_2) + (\Theta_1 - \Theta_2)^{\top})A^{\top}\bar{g}\bar{g} = 0$.

Since with overwhelming probability A has rank k, it must hold that $(\Theta_1 - \Theta_2) + (\Theta_1 - \Theta_2)^{\top} = 0$, that is, it must hold that $(\Theta_1 - \Theta_2) \in \mathcal{H}$. By construction, both for honestly generated proofs $\bar{\Pi}$ and simulated proofs this difference is uniformly distributed in $\mathcal{H}$. □

Efficiency comparison

To prove that $\bar{\mathbf{\Phi}} \in \mathcal{L}_{A,gk}$, for some $A \leftarrow \mathcal{D}_{\ell,k}$ with a Groth-Sahai instantiation based on a (possibly unrelated) $\mathcal{D}_{\ell',k'}$-matrix DH problem using standard Groth-Sahai proofs, one would prove that the following equation has a solution for all $i = 1, \ldots, \ell$:

$$r_1\bar{u}_{1,i} + \cdots + r_k\bar{u}_{k;i} = \bar{\Phi}_i;$$

that is, one needs to prove that ℓ linear equations with k variables are satisfied. Therefore, according to Figures 3.1 and 3.2, the verifier must be given $k\ell'$ elements of $\bar{\mathbb{G}}$ for the commitments and $\ell k'$ elements of $\bar{\mathbb{G}}$ for the proof. On the other hand, proving $\bar{\mathbf{\Phi}} \in \mathcal{L}_{A,gk}$ using our approach requires ℓk elements of $\bar{\mathbb{G}}$, corresponding to the size of the proof of one quadratic equation.

Application example 1

The standard proof of membership in $\mathcal{L}_{A,gk}$, when $A \leftarrow 2 - \mathsf{Lin}$ based on the same assumption (with $\ell = \ell' = 3, k = k' = 2$), requires 12 group elements, while with our approach only 6 elements are required. This reduces the ciphertext size of one of the instantiations of [LY12] from 15 to 9 group elements.

Application example 2

With our results, we can also give a more efficient proof of correct opening of a Cramer-Shoup ciphertext. We briefly recall the Cramer-Shoup encryption scheme based on the $2 - \mathsf{Lin}$-Assumption [Sha07, HK07]. The public key consists of the description of some group $\bar{\mathbb{G}}$ and a tuple $(\bar{a}_1, \bar{a}_2, \bar{b}_1, \bar{b}_2, \bar{b}_3, \bar{b}_4, \bar{b}_5, \bar{b}_6) \in \bar{\mathbb{G}}^8$. Given a message $m \in \bar{\mathbb{G}}$, a ciphertext is constructed by picking random $r, s \leftarrow \mathbb{Z}_p$ and setting

$$\bar{\mathbf{c}} := r(\bar{a}_1, \bar{0}, \bar{g}, \bar{b}_5, \bar{b}_1 + \alpha\bar{b}_3) + s(\bar{0}, \bar{a}_2, \bar{g}, \bar{b}_6, \bar{b}_2 + \alpha\bar{b}_4) + (\bar{0}, \bar{0}, \bar{m}, \bar{0}, \bar{0}),$$

where α is the hash of some components of the ciphertext and possibly some label. To prove that a ciphertext opens to a (known) message m, subtract m from the third component of the ciphertext and prove membership in $\mathcal{L}_{A,gk}$, where $A\alpha$ is defined as:

$$A_\alpha := \begin{pmatrix} a_1 & 0 \\ 0 & a_2 \\ 1 & 1 \\ b_5 & b_6 \\ b_1+\alpha b_3 & b_2+\alpha b_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \alpha \end{pmatrix} \begin{pmatrix} a_1 & 0 \\ 0 & a_2 \\ 1 & 1 \\ b_5 & b_6 \\ b_1 & b_2 \\ b_3 & b_4 \end{pmatrix},$$

where a_i, b_i are the discrete logarithms of $\bar{a}_i, \bar{b}_i$ with respect to $\bar{g}$. Denote M_α, C, the two matrices of the right term of the previous equation such that $A_\alpha = M_\alpha C$. The matrix A_α depends on α and is different for each ciphertext, so it cannot be included in the crs. Instead, we include the matrix $\bar{U}_C := (C||C\mathbf{w}+\mathbf{z}_C)\bar{g}$ in the soundness setting and $\bar{U}_C := (C||C\mathbf{w})\bar{g}$ in the WI setting, for $\mathbf{z}_C \notin \mathsf{Im}(C)$, for instance $\mathbf{z}_C^\top := (0,0,0,0,1,0)$

To prove membership in $\mathcal{L}_{A,gk}$ as we explained, we would make the proof with respect to the crs $\bar{U}_\alpha := M_\alpha \bar{U}_C$. Clearly, if $\mathbf{z}^\top := (0,0,0,0,1), \bar{U}_\alpha = (A_\alpha||A_\alpha\mathbf{w}+\mathbf{z})\bar{g}$ in the soundness setting and $\bar{U}_\alpha = (A_\alpha||A_\alpha\mathbf{w})\bar{g}$ in the WI, as required. The resulting proof consists of 10 group elements, as opposed to 16 using standard Groth-Sahai proofs. This applies to the result of [FLM11], Section 3.

3.5.2 More efficient proofs of validity of ciphertexts

The techniques of the previous section can be extended to prove the validity of a ciphertext. More specifically, given $A \leftarrow \mathcal{D}_{\ell,k}$, and some vector $\mathbf{z} \in \mathbb{Z}_p^\ell$, $\mathbf{z} \notin \mathsf{Im}(A)$, we show how to give a more efficient proof of membership in the space:

$$\mathcal{L}_{A,\mathbf{z},gk} = \{\bar{\mathbf{c}} : \bar{\mathbf{c}} = A\mathbf{r}\bar{g} + \mathbf{z}\bar{m}\} \subset \bar{\mathbb{G}}^\ell,$$

where $(\mathbf{r};\bar{m}) \in \mathbb{Z}_p^k \times \bar{\mathbb{G}}$ is the witness. This is also a proof of membership in the subspace of $\bar{\mathbb{G}}^\ell$ spanned by the columns of A and the vector $\mathbf{z}$, but the techniques given in Section 3.5.1 do not apply. The reason is that part of the witness, $\bar{m}$, is in the group $\bar{\mathbb{G}}$ and not in $\mathbb{Z}_p$, while to compute the subgroup membership proofs as described in Section 3.5.1 all the witness has to be in $\mathbb{Z}_p$.

In a typical application, $\bar{\mathbf{c}}$ will be the ciphertext of some encryption scheme, in which case $\mathbf{r}$ will be the ciphertext randomness and $\bar{m}$ the message. Deciding membership in this space is trivial when $\mathsf{Im}(A)$ and $\mathbf{z}$ span all $\mathbb{Z}_p^\ell$, so in particular our result is meaningful when $\ell > k+1$ and, in particular, that there exists some non-zero vector $\mathbf{s} \in \mathbb{Z}_p^\ell$ such that $\mathbf{s} \notin \langle \mathsf{Im}(A), \mathbf{z}\rangle$.

The construction

Define $\mathcal{H} := \{H \in \mathbb{Z}_p^{(k+2)\times(k+2)} : H + H^\top = 0\}$.

Setup : The setup gets and outputs a bilinear group $gk = (p, \bar{\mathbb{G}}, \mathbb{T}, e, \bar{g}) \leftarrow \mathcal{SPG}(1^\lambda)$.

GenCRS : We define $\bar{U} = (\bar{\mathbf{u}}_1, \ldots, \bar{\mathbf{u}}_{k+2})$ as $(A||\mathbf{z}||A\mathbf{w})\bar{g}$, where $A \leftarrow \mathcal{D}_{\ell,k}, \mathbf{w} \leftarrow \mathbb{Z}_p^k$, and $\mathbf{s} \notin \langle \mathsf{Im}(A), \mathbf{z} \rangle$. The common reference string is $\mathsf{crs} := (\bar{U}, \mathbf{s}, \mathbf{z})$.

$\mathsf{SimGenCRS}$: We define $\bar{U} = (\bar{\mathbf{u}}_1, \ldots, \bar{\mathbf{u}}_{k+2})$ as $(A||\mathbf{z}||A\mathbf{w} + \mathbf{s})\bar{g}$, where $A \leftarrow \mathcal{D}_{\ell,k}, \mathbf{w} \leftarrow \mathbb{Z}_p^k$, and $\mathbf{s} \notin \langle \mathsf{Im}(A), \mathbf{z} \rangle$. The common reference string is $\mathsf{crs} := (\bar{U}, \mathbf{s}, \mathbf{z})$. The simulation trapdoor τ is the vector $\mathbf{w} \in \mathbb{Z}_p^k$.

Prove : On input crs, a vector $\bar{\mathbf{c}} = A\mathbf{r}\bar{g} + \mathbf{z}\bar{m} \in \mathcal{L}_{A,\mathbf{z},gk}$ and the witness $(\mathbf{r}, \bar{m}) \in \mathbb{Z}_p^k \times \bar{\mathbb{G}}$, the prover chooses a matrix $H \leftarrow \mathcal{H}$ and computes

$$\bar{\Pi} = \mathbf{s}(\mathbf{r}^\top \bar{g}, \bar{m}, 0) + \bar{U}H.$$

$\mathsf{VerifyProof}$: On input $\mathsf{crs}, \bar{\mathbf{c}}, \bar{\Pi}$, the verifier checks if

$$\bar{\mathbf{c}}(\mathbf{s}\bar{g})^\top + (\mathbf{s}\bar{g})\bar{\mathbf{c}}^\top = \bar{\Pi}\bar{U}^\top + \bar{U}\bar{\Pi}^\top.$$

$\mathsf{SimProve}$: On input $\mathsf{crs}, \bar{\mathbf{c}}, \tau$ the simulator picks a matrix $H_{\mathsf{sim}} \leftarrow \mathcal{H}$ and computes

$$\bar{\Pi}_{\mathsf{sim}} = \bar{\mathbf{c}}(\mathbf{w}^\top, 0, -1) + \bar{U}H_{\mathsf{sim}}.$$

Theorem 3.5.2. *The proof system given above is a Non-Interactive Zero-Knowledge Proof for the language $\mathcal{L}_{A,\mathbf{z},gk}$, with perfect completeness, perfect soundness and composable zero-knowledge of $(k+2)\ell$ group elements based on the $\mathcal{D}_{\ell,k}$-MDDH Assumption.*

Proof. First, it is clear that under the $\mathcal{D}_{\ell,k} - \mathsf{MDDH}$ Assumption, the soundness (when crs is generated by GenCRS) and the WI setting (when crs is generated by $\mathsf{SimGenCRS}$) are computationally indistinguishable.

Completeness. To see completeness, note that by definition

$$\bar{\mathbf{c}} = \bar{U} \begin{pmatrix} \mathbf{r} \\ m \\ 0 \end{pmatrix},$$

where $m \in \mathbb{Z}_p$ is such that $\bar{m} = m\bar{g}$. Therefore, the left term of the verification equation is:

$$\bar{\mathbf{c}}(\mathbf{s}\bar{g})^\top + (\mathbf{s}\bar{g})\bar{\mathbf{c}}^\top = \bar{U} \begin{pmatrix} \mathbf{r}\bar{g} \\ \bar{m} \\ \bar{0} \end{pmatrix} \mathbf{s}^\top + \mathbf{s}\bar{U}(\mathbf{r}^\top \bar{g}, \bar{m}, \bar{0})$$

while the right term of a verification equation for a honestly created proof can be rewritten as:

$$\bar{\Pi}\bar{U}^\top + \bar{U}\bar{\Pi}^\top = \mathbf{s}(\mathbf{r}^\top \bar{g}, \bar{m}, \bar{0})\bar{U}^\top + \bar{U} \begin{pmatrix} \mathbf{r}\bar{g} \\ \bar{m} \\ \bar{0} \end{pmatrix} \mathbf{s}^\top + \bar{U}(H + H^\top)\bar{U}^\top$$

Since $H \in \mathcal{H}$, perfect completeness follows.

Soundness. Let $\boldsymbol{\xi} \in \mathbb{Z}_p^\ell$ be any vector such that $\boldsymbol{\xi}^\top A = \mathbf{0}$ and $\boldsymbol{\xi}^\top \mathbf{s} = 1$. If $\bar{\Pi}$ is any proof that satisfies the verification equation, multiplying on the left by $\boldsymbol{\xi}^\top$ and the right by $\boldsymbol{\xi}$ in the soundness setting,

$$\boldsymbol{\xi}^\top(\bar{\mathbf{c}}(\mathbf{s}\bar{g})^\top + (\mathbf{s}\bar{g})\bar{\mathbf{c}}^\top)\boldsymbol{\xi} = \boldsymbol{\xi}^\top(\bar{\Pi}\bar{U}^\top + \bar{U}\bar{\Pi}^\top)\boldsymbol{\xi},$$

we obtain

$$\boldsymbol{\xi}^\top \bar{\mathbf{c}}\bar{g} + \bar{g}\bar{\mathbf{c}}^\top \boldsymbol{\xi} = 2\boldsymbol{\xi}^\top \bar{\mathbf{c}}\bar{g} = 0_{\mathbb{T}}$$

Since this holds for any vector $\boldsymbol{\xi}$ such that $\boldsymbol{\xi}^\top A = \mathbf{0}$ and $\boldsymbol{\xi}^\top \mathbf{s} = 1$, this implies that $\bar{\mathbf{c}} \in \mathcal{L}_{A,\mathbf{z},gk}$, which proves perfect soundness.

Composable Zero-Knowledge. We will now see that, in the witness indistinguishability setting, both a real proof and a simulated proof have the same distribution when $\bar{\mathbf{c}} \in \mathcal{L}_{A,\mathbf{z},gk}$. We first note that they both satisfy the verification equation. Indeed, the left term of the equation in the WI setting is the same as before and obviously equal to the right term of the equation for the real proof. On the other hand, if $\bar{\mathbf{c}} \in \mathcal{L}_{A,\mathbf{z},gk}$, the right term of the verification equation for a simulated proof is:

$$\bar{\Pi}_{\mathsf{sim}}\bar{U}^\top + \bar{U}\bar{\Pi}_{\mathsf{sim}}^\top = \bar{\mathbf{c}}(\mathbf{w}^\top, 0, -1)\bar{U}^\top + \bar{U}\begin{pmatrix}\mathbf{w}^\top \\ 0 \\ -1\end{pmatrix}\bar{\mathbf{c}}^\top = \bar{\mathbf{c}}(\mathbf{s}\bar{g})^\top + (\mathbf{s}\bar{g})\bar{\mathbf{c}}^\top.$$

We now argue that an honestly generated proof $\bar{\Pi}$ and a simulated proof $\bar{\Pi}_{\mathsf{sim}}$ have the same distribution. By construction, there exist some matrices Θ and Θ' such that $\bar{\Pi} = \bar{U}\Theta$ and $\bar{\Pi}_{\mathsf{sim}} = \bar{U}\Theta'$. Now, if $\bar{\Pi}_1 = \bar{U}\Theta_1$ and $\bar{\Pi}_2 = \bar{U}\Theta_2$ are two proofs, real or simulated, which satisfy the verification equation, then necessarily $(\bar{\Pi}_1 - \bar{\Pi}_2)\bar{U}^\top + \bar{U}(\bar{\Pi}_1 - \bar{\Pi}_2) = \bar{U}((\Theta_1 - \Theta_2) + (\Theta_1 - \Theta_2)^\top)\bar{U}^\top = 0$.

Since with overwhelming probability, $\bar{U}$ has rank $k+2$, it must hold that $(\Theta_1 - \Theta_2) + (\Theta_1 - \Theta_2)^\top = 0$, that is, it must hold that $(\Theta_1 - \Theta_2) \in \mathcal{H}$. By construction, both for honestly generated proofs $\bar{\Pi}$ and simulated proofs $\bar{\Pi}_{\mathsf{sim}}$ this difference is uniformly distributed in $\mathcal{H}$. □

Efficiency comparison

It is straightforward to see that our proof requires $\ell(k+2)$ group elements. For sake of simplicity, let us assume $\mathbf{z}^\top = (0, \ldots, 0, 1) \in \mathbb{Z}_p^\ell$. There are two possible approaches to prove ciphertext validity based on a $\mathcal{D}_{\ell',k'}$- matrix assumption. In the first one, one commits to $\mathbf{r}, \bar{m}$ (which requires $\ell'(k+1)$ group elements) and then one proves that

$$\bar{\mathbf{c}}_{\ell-1} = (A_{\ell-1}\bar{g})\mathbf{r}, \qquad \bar{c}_\ell = (A_\ell \bar{g})\mathbf{r} + \bar{m},$$

where $\bar{\mathbf{c}}_{\ell-1}, A_{\ell-1}$ denote the first $\ell - 1$ elements of $\bar{\mathbf{c}}$ and the first $\ell - 1$ rows of A and $\bar{c}_\ell, A_\ell$ the last element and the last row. In this case, for the proof we need to give $k'(\ell - 1)$ elements for the first $\ell - 1$ equations and $\ell'(k+1)$ for the last equation (although the last equation is also linear, the witness is in $\mathbb{Z}_p$ and $\bar{\mathbb{G}}$ so the proof is the same size of a quadratic equation, see Figures 3.1 and 3.2). The second approach is to write the statement as a pairing product equation, in which case the prover commits to $\mathbf{r}\bar{g}, \bar{m}$ (which requires $\ell'(k+1)$ group elements) and it proves that:

$$\bar{\mathbf{c}}_{\ell-1}\bar{g} = (A_{\ell-1}\bar{g})(\mathbf{r}\bar{g}), \qquad \bar{\mathbf{c}}_{\ell}\bar{g} = (A_{\ell}\bar{g})(\mathbf{r}\bar{g}) + \bar{m}\bar{g}.$$

Now, all the equations are linear pairing product equations and the total cost of the proof is $\ell(k'+1)$. Therefore, in total we need $\ell'(k+1) + \ell'(k+1) + k'(\ell-1)$ group elements using the first approach and $\ell'(k+1) + \ell(k'+1)$ using the second. When adequate ℓ, k parameters are chosen, our proofs are therefore more efficient than the two mentioned alternative approaches.

Application example 1

We can use our results to show that the $2-\mathsf{Lin}$-based Cramer-Shoup encryption scheme described is a well formed ciphertext for some (secret) message $\bar{m}$. Let A_α, M_α, C the matrices described in the second example of Section 3.5.1. To apply our results in this section, include in the crs the matrix $\bar{U}_C := (C||\mathbf{z}_C||C\mathbf{w} + \mathbf{s}_C)\bar{g}$ in the soundness setting and $\bar{U}_C := (C||\mathbf{z}||C\mathbf{w})\bar{g}$ in the WI setting, where $\mathbf{s}_C^\top := (0,0,0,0,1,0)$ and $\mathbf{z}_C^\top := (0,0,1,0,0,0)$. To prove that a ciphertext is valid for a certain value of α, we would proceed as we just described with respect to the crs $\bar{U}_\alpha = M_\alpha \bar{U}_C$. In this application, in our case the size of the proof is of 20 group elements ($\ell = 5, k = 2$), while a proof of ciphertext validity based on $2-\mathsf{Lin}$ would require 24 group elements using the most efficient of the two approaches for these parameters ($\ell' = 3, k = 2, k' = 2, \ell = 5$).

3.5.3 More efficient proofs of plaintext equality

One of the applications of $\mathcal{D}_{\ell,k}$-MDDH assumptions given in [EHK+17] is building encryption schemes. One can define an encryption scheme which encryption algorithm works as follows. For a public key defined as $pk_A = (\bar{\mathbb{G}}; (A\bar{g}) \in \bar{\mathbb{G}}^{\ell\times k})$, for some $A \leftarrow \mathcal{D}_{\ell,k}$, given $\mathbf{r} \in \mathbb{Z}_p^k$, the encryption of $\bar{m} \in \bar{\mathbb{G}}$ is defined as:

$$\mathsf{Enc}(gk, pk_A, \bar{m}; \mathbf{r}) = \bar{\mathbf{c}} = A\mathbf{r}\bar{g} + (\bar{0}, \ldots, \bar{0}, \bar{m})^\top.$$

In [EHK+17], the decryption algorithm is also given and a reduction to the $\mathcal{D}_{\ell,k}$-MDDH is shown.

This encryption scheme corresponds to a commitment in Groth-Sahai proofs, with some particular randomness. Indeed, we can rewrite

$$\bar{\mathbf{c}} = A\mathbf{r}\bar{g} + (\bar{0}, \ldots, \bar{0}, \bar{m})^\top = A\mathbf{r}\bar{g} + \bar{m}\mathbf{z},$$

where $\mathbf{z} := (0, \ldots, 0, 1)^\top$. Note that $\bar{\mathbf{c}}$ can be seen as a commitment to $\bar{m}$ using randomness $\mathbf{s}^\top := (\mathbf{r}^\top, 0)$. Therefore, given two (potentially distinct) matrix distributions $\mathcal{D}_{\ell_1,k_1}, \mathcal{D}'_{\ell_2,k_2}$ and $A \leftarrow \mathcal{D}_{\ell_1,k_1} B \leftarrow \mathcal{D}'_{\ell_2,k_2}$, proving equality of plaintexts of two ciphertexts encrypted under pk_A, pk_B, corresponds to proving that two commitments under different keys open to the same value. Our proof will be more efficient because we do not give any commitments as part of the proof, since the ciphertexts themselves play this role. More specifically, given $\bar{\mathbf{c}}_A = \mathsf{Enc}(gk, pk_A, \bar{m}; \mathbf{r})$ and $\bar{\mathbf{c}}_B = \mathsf{Enc}(gk, pk_B, \bar{m}; \mathbf{s})$ we will treat $\bar{\mathbf{c}}_A$ as a commitment to the variable $\bar{x} \in \bar{\mathbb{G}} = \mathfrak{A}_1$ and $\bar{\mathbf{c}}_B$ as a commitment to the variable $\bar{y} \in \bar{\mathbb{G}} = \mathfrak{A}_2$ and prove that the quadratic equation $e(\bar{x}, \bar{g}) \cdot e(-\bar{g}, \bar{y}) = 0_{\mathbb{T}}$ is satisfied. The zero-knowledge simulator will open $\iota_1(\bar{g}), \iota_2(-\bar{g})$ as commitments to the $\bar{0}$ variable and simulate a proof for the equation $e(\bar{x}, \bar{0}) \cdot e(\bar{0}, \bar{y}) = 0_{\mathbb{T}}$, which is trivially satisfiable and can be simulated.

More formally, let

$$\mathcal{L}_{A,B,\mathbf{z}_1,\mathbf{z}_2,gk} := \{(\bar{\mathbf{c}}_A; \bar{\mathbf{c}}_B) : \bar{\mathbf{c}}_A = A\mathbf{r}\bar{g} + \bar{m}\mathbf{z}_1; \bar{\mathbf{c}}_B = B\mathbf{s}\bar{g} + \bar{m}\mathbf{z}_2\} \subset \bar{\mathbb{G}}^{\ell_1} \times \bar{\mathbb{G}}^{\ell_2}$$

where $\mathbf{r} \in \mathbb{Z}_p^{k_1}, \mathbf{s} \in \mathbb{Z}_p^{k_2}, \bar{g} \in \bar{\mathbb{G}}, \mathbf{z}_1 \in \mathbb{Z}_p^{\ell_1}$, and $\mathbf{z}_1 \notin \mathsf{Im}(A)$ and $\mathbf{z}_2 \in \mathbb{Z}_p^{\ell_2}, \mathbf{z}_2 \notin \mathsf{Im}(B)$

We now give

The construction

Setup : The setup specifies some group $gk = (p, \bar{\mathbb{G}}, \mathbb{T}, e, \bar{g}) \leftarrow \mathcal{SPG}(1^\lambda)$.

GenCRS : The common reference string crs specifies $\bar{U} = (\bar{\mathbf{u}}_1, \ldots, \bar{\mathbf{u}}_{k_1+1})$ and $\bar{V} = (\bar{\mathbf{v}}_1, \ldots, \bar{\mathbf{v}}_{k_2+1})$, which are $(\bar{U}, \bar{V}) = ((A||A\mathbf{w}_1)\bar{g}, (B||B\mathbf{w}_2)\bar{g})$, where $\mathbf{w}_1 \leftarrow \mathbb{Z}_p^{k_1}, \mathbf{w}_2 \leftarrow \mathbb{Z}_p^{k_2}, \mathbf{z}_1 \in \mathbb{Z}_p^{\ell_1}, \mathbf{z}_1 \notin \mathsf{Im}(A)$ and $\mathbf{z}_2 \in \mathbb{Z}_p^{\ell_2}, \mathbf{z}_2 \notin \mathsf{Im}(B)$. The common reference string is $\mathsf{crs} := (\bar{U}, \bar{V}, \mathbf{z}_1, \mathbf{z}_2)$.

$\mathsf{SimGenCRS}$: The common reference string crs specifies $\bar{U} = (\bar{\mathbf{u}}_1, \ldots, \bar{\mathbf{u}}_{k_1+1})$ and $\bar{V} = (\bar{\mathbf{v}}_1, \ldots, \bar{\mathbf{v}}_{k_2+1})$, which are $(\bar{U}, \bar{V}) = ((A||A\mathbf{w}_1 - \mathbf{z}_1)\bar{g}, (B||B\mathbf{w}_2 - \mathbf{z}_2)\bar{g})$, where $\mathbf{w}_1 \leftarrow \mathbb{Z}_p^{k_1}, \mathbf{w}_2 \leftarrow \mathbb{Z}_p^{k_2}, \mathbf{z}_1 \in \mathbb{Z}_p^{\ell_1}, \mathbf{z}_1 \notin \mathsf{Im}(A)$ and $\mathbf{z}_2 \in \mathbb{Z}_p^{\ell_2}, \mathbf{z}_2 \notin \mathsf{Im}(B)$. The crs is $\mathsf{crs} := (\bar{U}, \bar{V}, \mathbf{z}_1, \mathbf{z}_2)$. The trapdoor is $\tau = (\mathbf{w}_1, \mathbf{w}_2) \in \mathbb{Z}_p^{k_1} \times \mathbb{Z}_p^{k_2}$.

Prove : On input the setup gk, the common reference string crs, $(\bar{\mathbf{c}}_A, \bar{\mathbf{c}}_B) \in \mathcal{L}_{A,B,\mathbf{z}_1,\mathbf{z}_2,gk}$ and the witness $(\mathbf{r}, \mathbf{s}) \in \mathbb{Z}_p^{k_1} \times \mathbb{Z}_p^{k_2}$, pick $T \leftarrow \mathbb{Z}_p^{(k_1+1)\times(k_2+1)}$ and return

$$\bar{\Pi} = \mathbf{z}_2((\mathbf{r}\bar{g})^\top, \bar{0}) - \bar{V}T^\top \in \bar{\mathbb{G}}^{\ell_2 \times (k_1+1)}$$
$$\bar{\Theta} = -\mathbf{z}_1((\mathbf{s}\bar{g})^\top, 0) + \bar{U}T \in \bar{\mathbb{G}}^{\ell_1 \times (k_2+1)}.$$

$\mathsf{VerifyProof}$: On input $gk, \mathsf{crs}, \bar{\mathbf{c}}_A, \bar{\mathbf{c}}_B$ the verifier checks if

$$\bar{\mathbf{c}}_A(\mathbf{z}_2\bar{g})^\top - (\mathbf{z}_1\bar{g})\bar{\mathbf{c}}_B^\top = \bar{U}\bar{\Pi}^\top + \bar{\Theta}\bar{V}^\top.$$

$\mathsf{SimProve}$: On input $gk, \mathsf{crs}, \bar{\mathbf{c}}_A, \bar{\mathbf{c}}_B, \tau = (\mathbf{w}_1, \mathbf{w}_2)$ the simulator picks $T \leftarrow \mathbb{Z}_p^{(k_1+1)\times(k_2+1)}$ and returns

$$\bar{\Pi}_{\mathsf{sim}} = -\bar{\mathbf{c}}_B(\mathbf{w}_1^\top, -1) - \bar{V}T^\top, \qquad \bar{\Theta}_{\mathsf{sim}} = \bar{\mathbf{c}}_A(\mathbf{w}_2^\top, -1) + \bar{U}T.$$

Theorem 3.5.3. *Let $\mathcal{D}_{\ell_1,k_1}$ and $\mathcal{D}'_{\ell_2,k_2}$ be two matrix distributions and let $A \leftarrow \mathcal{D}_{\ell_1,k_1}, B \leftarrow \mathcal{D}'_{\ell_2,k_2}$. There exists a Non-Interactive Zero-Knowledge Proof for the language $\mathcal{L}_{A,B,\mathbf{z}_1,\mathbf{z}_2,gk}$ of $\ell_1(k_2+1)+\ell_2(k_1+1)$ group elements with perfect completeness, perfect soundness and composable zero-knowledge based on the $\mathcal{D}_{\ell_1,k_1}-\mathsf{MDDH}$ and the $\mathcal{D}_{\ell_2,k_2}-\mathsf{MDDH}$ Assumption.*

Proof. First, it is clear that under the $\mathcal{D}_{\ell,k}-\mathsf{MDDH}$ Assumption, the soundness and the WI setting are computationally indistinguishable.

Completeness. Note that the ciphertexts can be written as:

$$\bar{\mathbf{c}}_A = \bar{U}\begin{pmatrix}\mathbf{r}\\0\end{pmatrix} + \bar{m}\mathbf{z}_1 \qquad \mathbf{c}_B = \bar{V}\begin{pmatrix}\mathbf{s}\\0\end{pmatrix} + \bar{m}\mathbf{z}_2$$

therefore, the left term of the equation is of the form:

$$\begin{aligned}\bar{\mathbf{c}}_A(\mathbf{z}_2\bar{g})^\top - (\mathbf{z}_1\bar{g})\bar{\mathbf{c}}_B^\top &= \bar{U}\begin{pmatrix}\mathbf{r}\\0\end{pmatrix}(\mathbf{z}_2\bar{g})^\top - (\mathbf{z}_1\bar{g})(\mathbf{s}^\top,0)\bar{V}^\top + \bar{m}(\mathbf{z}_1\mathbf{z}_2^\top - \mathbf{z}_1\mathbf{z}_2^\top)\\ &= \bar{U}\begin{pmatrix}\mathbf{r}\\0\end{pmatrix}(\mathbf{z}_2\bar{g})^\top - (\mathbf{z}_1\bar{g})(\mathbf{s}^\top,0)\bar{V}^\top,\end{aligned}$$

from which it can easily be seen that an honestly generated proof satisfies the verification equation.

Soundness. Let $\boldsymbol{\xi}_1 \in \mathbb{Z}_p^{\ell_1}$ be any vector such that $\boldsymbol{\xi}_1^\top A = \mathbf{0}, \boldsymbol{\xi}_1^\top \mathbf{z}_1 = 1$ and let $\boldsymbol{\xi}_2 \in \mathbb{Z}_p^{\ell_2}$ be any vector such that $\boldsymbol{\xi}_2^\top B = \mathbf{0}, \boldsymbol{\xi}_2^\top \mathbf{z}_2 = 1$. Let $\bar{\Pi}, \bar{\Theta}$ be any proof that satisfies the verification equation. In the soundness setting, the verification equation holds if and only if

$$\boldsymbol{\xi}_1^\top(\bar{\mathbf{c}}_A(\mathbf{z}_2\bar{g})^\top - (\mathbf{z}_1\bar{g})\bar{\mathbf{c}}_B^\top)\boldsymbol{\xi}_2 = \boldsymbol{\xi}_1^\top(\bar{U}\bar{\Pi}^\top + \bar{\Theta}\bar{V}^\top)\boldsymbol{\xi}_2,$$

from which it follows that

$$\boldsymbol{\xi}_1^\top\bar{\mathbf{c}}_A\bar{g} - \bar{g}\bar{\mathbf{c}}_B^\top\boldsymbol{\xi}_2 = 0_{\mathbb{T}}$$

which implies that $\boldsymbol{\xi}_1^\top\bar{\mathbf{c}}_A = \bar{\mathbf{c}}_B^\top\boldsymbol{\xi}_2$. Since this holds for any possible $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2$ meeting the aforementioned conditions, one can easily conclude that $\bar{m}_1 = \bar{m}_2$, as required.

Composable zero-knowledge. Clearly, an honestly generated proof also satisfies the verification equation in the WI setting. On the other hand, a simulated proof satisfies the verification equation, since,

$$\begin{aligned}\bar{\mathbf{c}}_A(\mathbf{z}_2\bar{g})^\top - (\mathbf{z}_1\bar{g})\bar{\mathbf{c}}_B^\top &= \bar{\mathbf{c}}_A(\bar{V}(\mathbf{w}_2^\top,-1)^\top)^\top - \bar{U}(\mathbf{w}_1^\top,-1)^\top\bar{\mathbf{c}}_B^\top\\ &= (\bar{\mathbf{c}}_A(\mathbf{w}_2^\top,-1) + \bar{U}T)\bar{V}^\top - \bar{U}((\mathbf{w}_1^\top,-1)^\top\bar{\mathbf{c}}_B^\top - T\bar{V}^\top)\end{aligned}$$

for any $T \leftarrow \mathbb{Z}_p^{(k_1+1)\times(k_2+1)}$. In the WI setting, $\bar{\mathbf{c}}_A, (\mathbf{z}_1\bar{g}) \in \langle\bar{\mathbf{u}}_1;\ldots;\bar{\mathbf{u}}_{k_1+1}\rangle$ and $\bar{\mathbf{c}}_B, (\mathbf{z}_2\bar{g}) \in \langle\bar{\mathbf{v}}_1,\ldots,\bar{\mathbf{v}}_{k_2+1}\rangle$. In either case (real or simulated), the matrix T is random, so we can think of $\bar{\Theta}$ as uniformly distributed. On the other hand, for any fixed $\bar{\Theta}$, any two proofs $\bar{\Pi}, \bar{\Pi}'$ which satisfy the verification equation, it must hold that $0_{\mathbb{T}} = \bar{U}(\bar{\Pi} - \bar{\Pi}')^\top$. In the WI setting, since $\bar{\mathbf{c}}_B \in \langle\bar{\mathbf{v}}_1,\ldots,\bar{\mathbf{v}}_{k_2+1}\rangle$, both for the real and the simulated proof this equation holds if and only if there exists a non-zero matrix H such that$0_{\mathbb{T}} = \bar{U}H\bar{V}^\top$. Such a matrix does not exist because in the WI setting $\bar{U}$ and $\bar{V}$ have full rank. Therefore, both for real and simulated proofs, $\bar{\Theta}$ and $\bar{\Pi}$ are uniformly distributed among all the proofs that satisfy the verification equation. □

Efficiency comparison

The size of our proof of membership in $(\bar{\mathbf{c}}_A, \bar{\mathbf{c}}_B) \in \mathcal{L}_{A,B,\mathbf{z}_1,\mathbf{z}_2,gk}$ for some $A \leftarrow \mathcal{D}_{\ell_1,k_1}, B \leftarrow \mathcal{D}'_{\ell_2,k_2}$, is $\ell_1(k_2+1) + \ell_2(k_1+1)$. The size of a standard proof depends on general of the specific $A, B, \mathbf{z}_1, \mathbf{z}_2$. We discuss several examples of applications below.

Example of application 1

When $\ell_1 = k_1+1, \ell_2 = k_2+1, \mathbf{z}_1^\top = (0,\ldots,0,1) \in \mathbb{Z}_p^{k_1+1}$ and $\mathbf{z}_2^\top = (0,\ldots,0,1) \in \mathbb{Z}_p^{k_2+1}$ the natural approach is the one described in [KV13], namely, if we denote by A_0 and $\bar{\mathbf{c}}_{A,0}$ the first k_1 rows of A and the first k_1 elements of $\bar{\mathbf{c}}_A$, A_1 and $\bar{\mathbf{c}}_{A,1}$ the last row and element, by B_0 and $\bar{\mathbf{c}}_{B,0}$ the first k_2 rows of B and the first k_2 elements of $\bar{\mathbf{c}}_B$ and $B_1, \bar{\mathbf{c}}_{B,1}$ the last row and element, one would prove that the following equations are satisfied

$$(A_0\bar{g})\mathbf{x} = \bar{\mathbf{c}}_{A,0}$$
$$(B_0\bar{g})\mathbf{y} = \bar{\mathbf{c}}_{B,0}$$
$$(A_1\bar{g})\mathbf{x} - (B_1\bar{g})\mathbf{y} = \bar{\mathbf{c}}_{A,1} - \bar{\mathbf{c}}_{B,1}.$$

That is, one needs to prove that (k_1+k_2+1) linear multiscalar multiplication equations are satisfiable with k_1+k_2 variables. Therefore, if one uses an instantiation of Groth-Sahai based on some $\mathcal{D}_k - \mathsf{MDDH}$ problem, one needs $(k_1+k_2+1)k$ group elements for the proof and $(k_1+k_2)(k+1)$ group elements for the commitments, whereas in our case we give a total of $2(k_1+1)(k_2+1)$ group elements. For the special case of $2-\mathsf{Lin}$, with $A \leftarrow \mathcal{L}_2, B \leftarrow \mathcal{L}_2$ a proof based on the $2-\mathsf{Lin}$ instantiation of Groth-Sahai proofs as described in [KV13] requires thus 22 elements as opposed to 18. On the other hand, for the encryption scheme of Hofheinz and Jager with tight security reduction to 2-Lin [HJ16] we need to make a proof for equality of plaintexts written as a pairing product equation. This is because the authors need to convert an OR of sets of pairing product equations into an AND of pairing product equations. In terms of pairing product equations, equality of plaintexts is expressed as

$$e((A_0\bar{g})\mathbf{x}, \bar{g}) = e(\bar{\mathbf{c}}_{A,0}, \bar{g})$$
$$e((B_0\bar{g})\mathbf{y}, \bar{g}) = e(\mathbf{c}_{B,0}, \bar{g})$$
$$e((A_1\bar{g})\mathbf{x} - (B_1\bar{g})\mathbf{y}, \bar{g}) = e(\bar{\mathbf{c}}_{A,1} - \bar{\mathbf{c}}_{B,1}, \bar{g})$$

and requires longer proofs, namely $(k_1 + k_2 + 1)(k + 1)$ group elements for the proof and $(k_1 + k_2)(k + 1)$ for the commitments. In this case, for $2 - \mathsf{Lin}$ the proof is reduced from 27 elements to 18 elements.

More general relations

We think our results can be extended to give more efficient arguments of knowledge for affine relations among ciphertexts encrypted under more than two different public keys. More specifically, given some ordered sequence of matrices $A_1, \ldots, A_n$ independently sampled from $\mathcal{D}^1_{\ell_1,k_1}, \ldots, \mathcal{D}^n_{\ell_n,k_n}$, some vectors $\mathbf{z}_i \leftarrow \mathbb{Z}_p^\ell, \mathbf{z}_i \notin \mathsf{Im}(A_i)$, and some ciphertexts $\bar{\mathbf{c}}_i = (A_i\bar{g})\mathbf{r}_i + \bar{m}_i\mathbf{z}_i$, one could use similar techniques to prove that the ciphertexts are such that $\sum_{i=1}^n b_i\bar{m}_i = \bar{t}$, for some constants $b_1, \ldots, b_n \in \mathbb{Z}_p$ and $\bar{t} \in \bar{\mathbb{G}}$.

3.5.4 Commitment schemes

The languages for which we have given more efficient proofs in this section arise naturally when one tries to prove statements about ciphertexts, but they naturally extend to more general commitment schemes (obviously a ciphertext can be seen as a special type of commitment). For instance, in [FLM11], the authors prove that a Cramer-Shoup ciphertext encrypts the same value as a Groth-Sahai commitment based on $2 - \mathsf{Lin}$. To get a more efficient proof for this statement or also for the statement that two Groth Sahai commitments based on different matrix assumptions correspond to the same value, we can essentially use the same approach as the one of Section 3.5.3.

Chapter 4

Fine-Tuning Groth-Sahai Proofs

This chapter is mainly based on the results published in [EG14], coauthored with Jens Groth.

4.1 Our contributions

In this contribution, we focus on improving Groth-Sahai proofs efficiency and we propose several ways to fine-tune Groth-Sahai zero-knowledge proofs in the asymmetric SXDH bilinear group setting, which is the one that yields the most efficient NIZK proofs.

- Groth-Sahai proofs use public constants and committed variables. We introduce two new types of values: public base elements and encrypted variables. This reduces the size of proofs for statements involving these values.
- We recast Groth-Sahai proofs as a commit-and-prove scheme. This makes it possible to reuse commitments in the proofs of different statements even when these statements depend on previous commitments and proofs.
- We show that the prover's computation can be reduced by letting her pick her own provably correct common reference string.

Encrypted variables

The common reference string in Groth-Sahai proofs contains a public commitment key that the prover uses to commit to variables. The prover then proceeds to prove that the committed variables satisfy the equations in the statement. The commitment key can be set up to be perfectly binding, in which case it is impossible to cheat in the proof (perfect soundness). Alternatively, the commitment key can be set up to be perfectly hiding, in which case the NIZK proof can be simulated without knowledge of the witness (perfect zero-knowledge).

In our scheme, we allow the prover to encrypt variables using ElGamal encryption as an alternative to the commitment scheme. ElGamal encryption requires two scalar multiplications and commitments in Groth-Sahai proofs require four scalar multiplications so this reduces the prover's computation. Moreover, equations that use ElGamal ciphertexts instead of commitments have simpler proofs and we save two group elements for each of those equations.

We must be careful when using ElGamal encryption though since it will always be perfectly binding regardless of the key. We can therefore not set up the common reference string to give us perfect zero-knowledge any more. Instead, we rely on the Decision Diffie-Hellman (DDH) assumption to get computational zero-knowledge and we place some restrictions on the types of equations where ElGamal encryptions can be used.

Base elements

Groth-Sahai proofs are zero-knowledge if all pairing-product equations have $t_{\mathbb{T}} = 0_{\mathbb{T}}$ and no public constants are paired with each other. Groth and Sahai gave a technique to rewrite the equations to enable zero-knowledge proofs when $t_{\mathbb{T}} = \sum_{i=1}^{n} e(\hat{t}_i, \check{t}_i)$, however, this comes at a cost of $O(n)$ group elements.

We observe that the commitment keys can be set up to allow simulation in the special case where the public constants are the base elements $\hat{g}$ or $\check{h}$. In the context of Groth-Sahai proofs this makes zero-knowledge simulation possible when $t_{\mathbb{T}} = e(\hat{a}, \check{h}) + e(\hat{g}, \check{b})$ for public constants $\hat{a} \in \hat{\mathbb{G}}$ and $\check{b} \in \check{\mathbb{H}}$. This extension of Groth-Sahai proofs comes at no extra cost, so we save the costly rewriting of the equations to get zero-knowledge.

In addition, Groth-Sahai zero-knowledge proofs for multi-scalar multiplications equations in $\hat{\mathbb{G}}$ or in $\check{\mathbb{H}}$ in which all the field elements are constants consist of 6 group elements unless $\hat{t} = \hat{0}$ or $\check{t} = \check{0}$, in which case the size of the proof is reduced to 2 group elements. We observe that if $\hat{t} = \hat{g}$ or $\check{t} = \check{h}$ then we can still have zero-knowledge proofs which consist of 2 group elements.

Using commitment keys with known discrete logarithms

In Groth-Sahai proofs, the prover uses a common reference string shared between the prover and the verifier to create NIZK proofs. This common reference string is created by a trusted entity and, in particular, the discrete logarithms of the commitment keys with respect to $\hat{g}$ and $\check{h}$ are unknown to both the prover and the verifier.

In this contribution, we show how to reduce the prover's computation by allowing her to choose her *own common reference string*, which we think of as her public key. This change reduces the cost of computing her commitments from 4 scalar multiplications to 2 scalar multiplications and it also reduces the cost of computing proofs.

The verifier does of course need a guarantee that the prover's public key is formed as a perfectly binding common reference string, otherwise the proofs would not be sound any more. This can be accomplished by letting the prover give a Groth-Sahai proof, using a common reference string the verifier does trust, for the public key being correct. We show that the prover can communicate her public key and prove it is correct using 12 group elements in total, which is a one-off cost as the public key can be used for many commitments and proofs.

Seeing the common reference string as the prover's public key also gives us some flexibility in the setup. Instead of proving the public key correct in the common reference string model, the prover could use the multi-string model [GO14] where the setup assumption is relaxed to assuming a majority out of n common reference strings are honest. Alternatively, the prover could give a zero-knowledge proof of knowledge to a trusted third party that the public key is correct and get a certificate on the public key.

Type-based commit-and-prove schemes

A commit-and-prove scheme [Kil90, CLOS02] is a natural generalization of a zero-knowledge proof, where the prover can commit to values and prove statements about the committed values. Commit-and-prove schemes provide extra flexibility and reduce communication; the same commitments can for instance be used in different proofs for adaptively chosen statements about the committed values.

Groth-Sahai proofs can be used to build a non-interactive commit-and-prove scheme in a natural way. Fuchsbauer [Fuc11] defined a witness-indistinguishable Groth-Sahai based commit-and-prove scheme and used it in the construction of delegatable anonymous credentials.

We will formulate our construction as a non-interactive commit-and-prove scheme because it allows for greater flexibility. It is for instance possible to choose values to be committed to in an adaptive fashion that depends on previous commitments or proofs, while the traditional definition of NIZK proofs would require the prover to make an entirely new set of commitments for each statement to be proven.

Our results are natural extensions of Groth-Sahai proofs and our definition of a non-interactive commit-and-prove scheme will resemble Fuchsbauer's [Fuc11]. However, we are in a different situation because we have more types of elements that we want to commit to. A group element in $\hat{\mathbb{G}}$ may for instance be committed using the perfectly binding/perfectly hiding commitment scheme or using ElGamal encryption. We could resolve this by using multiple commitment schemes, i.e., define and construct a many-commitment-scheme-and-prove system. However, the definition would quickly become unwieldy as the number of commitment schemes grows.

To give a generally applicable definition of non-interactive commit-and-prove schemes, we instead propose the notion of *type-based* commitments. A type-based commitment scheme enables the prover to commit to a message m with a publicly known type t. One example of a type could for instance be $t = \mathrm{enc}_{\hat{\mathbb{G}}}$ meaning the value m should be encrypted (as opposed to using the more expensive commitment operation) and it should be done in group $\hat{\mathbb{G}}$. Type-based commitments resemble tag-based commitments, where a message m can be committed under an arbitrary tag tag, but a crucial difference is that while tags can be arbitrary bit-strings, types may be restricted. More precisely, we require that the type and message pair (t, m) belong to a message space $\mathfrak{M}_{gk}$. We consider the type as being part of the message, so another way of looking at a type-based commitment scheme is to say a message consists of public part t and a private part m. This increases the flexibility of the commitment scheme, we can for instance create a type $(\text{pub}_{\hat{\mathbb{G}}}, x)$ that publicly declares the committed value x. Since the type is public this commitment is no longer hiding, however, as we shall see it simplifies our commit-and-prove scheme because we can now commit to both public constants and secret variables without having to treat them differently.

Armed with type-based commitments we provide a formal definition of commit-and-prove schemes, where the prover may commit to values and prove statements about the values. We stress that commitments and proofs can be arbitrarily interleaved so at any given point a new value or statement may depend on previous commitments and proofs, which makes them more flexible than standard Groth-Sahai proofs. The commit-and-prove scheme is defined to be sound if it is impossible to prove a false statement about the committed values and it is defined to be zero-knowledge if there exists a simulator that can simulate commitments and proofs.

Applications

To illustrate the advantages of our fine-tuned Groth-Sahai proofs we will give an example based on the weak Boneh-Boyen signature scheme [BB08], which is widely used in pairing-based protocols. The verification key is an element $\hat{v} \in \hat{\mathbb{G}}$ and a signature on a message $m \in \mathbb{Z}_p$ is a group element $\check{\sigma} \in \check{\mathbb{H}}$ such that

$$e(\hat{v} + m\hat{g}, \check{\sigma}) = e(\hat{g}, \check{h}).$$

Suppose the prover has commitments to $\hat{v}$ and $\check{\sigma}$ and wants to demonstrate that they satisfy the verification equation for a (public) message m. With traditional Groth-Sahai proofs the commitments c and d to $\hat{v}$ and $\check{\sigma}$ would be treated as part of the statement and one would carefully demonstrate the existence of openings of c and d to $\hat{v}$ and $\check{\sigma}$ satisfying the pairing-product equation. With a commit-and-prove system, we can instead jump directly to demonstrating that the values inside $\hat{v}$ and $\check{\sigma}$ satisfy the verification equation without having to treat the openings of the commitments as part of the witness. This saves several group elements each time one of the commitments is used.

Next, observe that the pairing-product equation has $t_{\mathbb{T}} = e(\hat{g}, \check{h})$. A direct application of Groth-Sahai proofs would therefore not yield a zero-knowledge proof but only give witness-indistinguishability. To get zero-knowledge we could use the workaround suggested by Groth-Sahai, which would consist of committing to a new variable $\check{y}$, prove that $\check{y} = \check{h}$ and simultaneously $e(\hat{v} + m\hat{g}, \check{\sigma}) - e(\hat{g}, \check{y}) = 0_{\mathbb{T}}$. This work-around would increase the cost of the proof from 8 group elements to 16 group elements, so we save 8 group elements by enabling a direct proof.

Now assume the prover has created her own common reference string pk and has already sent it together with the well-formedness proof to the verifier. The prover could now use pk to compute the zero-knowledge proof for the equation $(\hat{v} + m\hat{g}) \cdot \check{\sigma} = \hat{g}\check{h}$. By using pk, she would need to do 10 scalar multiplications in $\hat{\mathbb{G}}$ and 6 scalar multiplications in $\check{\mathbb{H}}$ to compute the proof. In contrast, if she was computing the proof using the commitment key ck, she would need to do 12 scalar multiplications in $\hat{\mathbb{G}}$ and 10 scalar multiplications in $\check{\mathbb{H}}$. As the operations in $\check{\mathbb{H}}$ are usually significantly more expensive than the operations in $\hat{\mathbb{G}}$, the prover is essentially saving 4 expensive operations of the 10 that she would need to do if she used ck. Therefore, our techniques reduce the computational cost of creating the zero-knowledge proof by roughly 40%. In addition, the computational cost of computing the commitments to $\hat{v}$ and $\check{\sigma}$ would also be reduced by 50%.

Finally, we can obtain a saving by encrypting one of the variables instead of committing to it. If we encrypt $\hat{v}$ for instance, the ciphertext is still 2 group elements just as the commitment in a Groth-Sahai proof would be, but the cost of the proof for the pairing-product equation is reduced from 8 group elements to 6 group elements. In total, we have reduced the cost of the proof by 63% from 16 group elements to 6 group elements.

In Sec. 4.9 we give two concrete examples of existing schemes using Groth-Sahai proofs where our techniques can improve efficiency.

4.2 Commit-and-prove scheme definitions

In Section 2.8 we explained how to define a setup-dependent language L_{gk} from a relation R containing tuples of setup, statement and witness (gk, x, w).

We will now define a commit-and-prove scheme for a relation R. In the commit-and-prove scheme, we may commit to different values $w_1, \ldots, w_N$ and prove for different statements x that a subset of the committed values $w = (w_{i_1}, \ldots, w_{i_n})$ constitute a witness for $x \in L_{gk}$, i.e., $(gk, x, w) \in R$.

We will divide each committed value into two parts $w_i = (t_i, m_i)$. The first part t_i can be thought of as a public part that does not need to be kept secret, while the second part m_i can be thought of as a secret value that our commit-and-prove scheme should not reveal. The first part t_i will be useful later on to specify the type of value m_i is, for instance a group element or a field element, and to specify which type of commitment we should make to m_i. This is a natural and useful generalization of standard commitment schemes.

A commit-and-prove scheme CP $= (\mathsf{Setup}, \mathsf{GenCRS}, \mathsf{Commit}, \mathsf{Prove}, \mathsf{VerifyProof})$ consists of five polynomial time algorithms. The algorithms $\mathsf{Setup}, \mathsf{GenCRS}, \mathsf{Prove}$ are probabilistic and the algorithms $\mathsf{Commit}, \mathsf{VerifyProof}$ are deterministic.

$\mathsf{Setup}(1^\lambda)$**:** On input a security parameter 1^λ outputs a setup gk. The setup specifies a message space $\mathfrak{M}_{gk}$, a randomness space $\mathfrak{R}_{gk}$ and a commitment space $\mathfrak{C}_{gk}$. Membership of either space can be decided efficiently.

$\mathsf{GenCRS}(gk)$**:** On input a setup gk generates a commitment key ck.

$\mathsf{Commit}(gk, ck, t, m; r)$**:** Given a setup gk, a commitment key ck, a message $(t, m) \in \mathfrak{M}_{gk}$ and randomness r such that $(t, r) \in \mathfrak{R}_{gk}$ returns a commitment c such that $(t, c) \in \mathfrak{C}_{gk}$.

$\mathsf{Prove}(gk, ck, x, (t_1, m_1, r_1), \ldots, (t_n, m_n, r_n))$**:** Given a setup gk, a commitment key ck, a statement x and commitment openings such that $(t_i, m_i) \in \mathfrak{M}_{gk}, (t_i, r_i) \in \mathfrak{R}_{gk}$ and $(gk, x, t_1, m_1, \ldots, t_n, m_n) \in R$ returns a proof π.

$\mathsf{VerifyProof}(gk, ck, x, (t_1, c_1), \ldots, (t_n, c_n), \pi)$**:** Given a setup gk, a commitment key ck, a statement x, a proof π and commitments $(t_i, c_i) \in \mathfrak{C}_{gk}$ returns 1 (accept) or 0 (reject).

Definition 4.2.1 (Perfect correctness)**.** *The commit-and-prove system* CP *is (perfectly) correct if for all adversaries $\mathcal{A}$*

$$\Pr\left[\begin{array}{l} gk \leftarrow \mathsf{Setup}(1^\lambda); ck \leftarrow \mathsf{GenCRS}(gk)\ ;\ (x, w_1, r_1, \ldots, w_n, r_n) \leftarrow \mathcal{A}(gk, ck)\ ; \\ c_i \leftarrow \mathsf{Commit}(gk, ck, w_i; r_i)\ ; \pi \leftarrow \mathsf{Prove}(gk, ck, x, w_1, r_1, \ldots, w_n, r_n)\ : \\ \mathsf{VerifyProof}(gk, ck, x, (t_1, c_1), \ldots, (t_n, c_n), \pi) = 1 \end{array}\right] = 1,$$

where $\mathcal{A}$ outputs w_i, r_i such that $w_i = (t_i, m_i) \in \mathfrak{M}_{gk}, (t_i, r_i) \in \mathfrak{R}_{gk}$ and $(gk, x, w_1, \ldots, w_n) \in R$.

We say a commit-and-prove scheme is sound if it is impossible to prove a false statement. Strengthening the usual notion of soundness, we will associate unique values to the commitments, and these values will constitute a witness for the statement. This means that not only does a valid proof guarantee the truth of the statement, but also each commitment will always contribute a consistent witness towards establishing the truth of the statement.

Definition 4.2.2 (Perfect soundness)**.** *The commit-and-prove system* CP *is (perfectly) sound if there exists a deterministic (unbounded) opening algorithm* Open *such that for all adversaries $\mathcal{A}$*

$$\Pr\left[\begin{array}{l} gk \leftarrow \mathsf{Setup}(1^\lambda); ck \leftarrow \mathsf{GenCRS}(gk)\ ; \\ (x, t_1, c_1, \ldots, t_n, c_n, \pi) \leftarrow \mathcal{A}(gk, ck)\ ;\ m_i \leftarrow \mathsf{Open}(gk, ck, t_i, c_i)\ : \\ \mathsf{VerifyProof}(gk, ck, x, t_1, c_1, \ldots, t_n, c_n, \pi) = 0\ \vee\ (gk, x, (t_1, m_1), \ldots, (t_n, m_n)) \in R \end{array}\right] = 1.$$

Extending the notion of soundness, we may define knowledge as the ability to efficiently extract a witness for the truth of the statement proven when given an extraction key xk. Actually, the commit-and-prove schemes we construct will not allow the extraction of all types of witnesses due to the hardness of the discrete logarithm problem. However, we can specify a function F such that we can extract $F(gk, w)$ from a commitment. Efficient extraction of a witness corresponds to the special case where $F(gk, w) = m$, with m being the secret part of the witness $w = (t, m)$.

Definition 4.2.3 (Perfect F-knowledge). *Let in the following* $\mathsf{ExtGenCRS}$ *and* $\mathsf{Extract}$ *be two algorithms as described below.*

- $\mathsf{ExtGenCRS}$ *is a p.p.t. algorithm that on gk returns (ck, xk). We call ck the commitment key and xk the extraction key. We require that the probability distributions of ck made by* $\mathsf{ExtGenCRS}$ *and* GenCRS *are identical.*
- $\mathsf{Extract}$ *is a deterministic polynomial time algorithm that given a setup gk, an extraction key xk and $(t, c) \in \mathfrak{C}_{gk}$ returns a value.*

The commit-and-prove scheme CP *with perfect soundness for opening algorithm* Open *has F-knowledge for a function F if for all adversaries $\mathcal{A}$*

$$\Pr\left[\begin{array}{l} gk \leftarrow \mathsf{Setup}(1^\lambda); (ck, xk) \leftarrow \mathsf{ExtGenCRS}(gk)\ ;\ (t, c) \leftarrow \mathcal{A}(gk, ck, xk) : \\ (t, c) \notin \mathfrak{C}_{gk} \vee \mathsf{Extract}(gk, xk, t, c) = F(gk, t, \mathsf{Open}(gk, ck, t, c)) \end{array}\right] = 1.$$

We say a commit-and-prove scheme is zero-knowledge if it does not leak information about the secret parts of the committed messages besides what is already leaked by the public parts. This is defined as the possibility to simulate commitments and proofs without knowing the secret parts of the messages (the types are known) if instead some secret simulation trapdoor is known about the commitment key ck.

Following [Gro06, GOS12] we define a strong notion of zero-knowledge called composable zero-knowledge. Composable zero-knowledge says the commitment key can be simulated, and if the commitment key is simulated it is not possible to distinguish real proofs from simulated proofs even if the simulation trapdoor is known.

Definition 4.2.4 (Composable zero-knowledge). *The commit-and-prove system* CP *is computationally composable zero-knowledge if there exist p.p.t. algorithms* $\mathsf{SimGenCRS}$, $\mathsf{SimCommit}$, $\mathsf{SimProve}$ *such that for all non-uniform polynomial time stateful interactive adversaries $\mathcal{A}$*

$$\begin{aligned} &\Pr\left[gk \leftarrow \mathsf{Setup}(1^\lambda); ck \leftarrow \mathsf{GenCRS}(gk) : \mathcal{A}(gk, ck) = 1\right] \\ \approx &\Pr\left[gk \leftarrow \mathsf{Setup}(1^\lambda); (ck, tk) \leftarrow \mathsf{SimGenCRS}(gk) : \mathcal{A}(gk, ck) = 1\right] \end{aligned}$$

and

$$\Pr\left[\begin{array}{l} gk \leftarrow \mathsf{Setup}(1^\lambda); (ck, tk) \leftarrow \mathsf{SimGenCRS}(gk); \\ (x, i_1, \ldots, i_n) \leftarrow \mathcal{A}^{\mathsf{Commit}(gk, ck, \cdot)}(gk, ck, tk); \\ \pi \leftarrow \mathsf{Prove}(gk, ck, x, w_{i_1}, r_{i_1}, \ldots, w_{i_n}, r_{i_n}) : \\ \mathcal{A}(\pi) = 1 \end{array}\right]$$

$$\approx \Pr\left[\begin{array}{l} gk \leftarrow \mathsf{Setup}(1^\lambda); (ck, tk) \leftarrow \mathsf{SimGenCRS}(gk); \\ (x, i_1, \ldots, i_n) \leftarrow \mathcal{A}^{\mathsf{SimCommit}(gk, tk, \cdot)}(gk, ck, tk); \\ \pi \leftarrow \mathsf{SimProve}(gk, tk, x, t_{i_1}, s_{i_1}, \ldots, t_{i_n}, s_{i_n}) : \\ \mathcal{A}(\pi) = 1 \end{array}\right],$$

where

- *tk is a trapdoor key used to construct simulated proofs*
- $\mathsf{Commit}(gk, ck, \cdot)$ *on* $w_i = (t_i, m_i) \in \mathfrak{M}_{gk}$ *picks uniformly random* r_i *such that* $(t_i, r_i) \in \mathfrak{R}_{gk}$ *and returns* $c_i = \mathsf{Commit}(gk, ck, w_i; r_i)$
- $\mathsf{SimCommit}(gk, tk, \cdot)$ *on* $w_i = (t_i, m_i) \in \mathfrak{M}_{gk}$ *runs* $(c_i, s_i) \leftarrow \mathsf{SimCommit}(gk, tk, t_i)$ *and returns* c_i*, where* s_i *is some auxiliary information used to construct simulated proofs*
- $\mathcal{A}$ *picks* $(x, i_1, \ldots, i_n)$ *such that* $(gk, x, w_{i_1}, \ldots, w_{i_n}) \in R$

Comparison to standard zero-knowledge proofs. The commit-and-prove functionality defined above is a generalization of standard zero-knowledge proofs. The commitment key corresponds to a common reference string. To give a proof for $x \in L_{gk}$ the prover could commit to the witness and then give a proof. However, note that languages appearing in standard zero-knowledge proofs are not parametrized by any setup. The correctness, soundness and zero-knowledge properties given above imply that this would yield a NIZK proof system for R.

The generalization to a commit-and-prove functionality is quite useful for efficiency reasons. Using a traditional zero-knowledge proof system the prover has to start from scratch every time a new statement has to be proved. In a commit-and-prove system, commitments to values can be reused in multiple proofs, which reduces both computation and communication.

There are also conceptual advantages to using a commit-and-prove functionality. One is that it is convenient that the same committed values can be used across multiple proofs. Another one is that we capture in a natural way that statements may be related to commitments and proofs that have been made before.

4.3 Preliminaries

4.3.1 Using linear algebra to express bilinear group equations

Let $(p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathcal{PG}$ be a type-III bilinear group.

Writing the pairing multiplicatively allows us to use linear algebra notation in a natural way, we have for instance

$$\hat{x} \begin{pmatrix} \check{0} & \check{y} \\ \check{z} & \check{0} \end{pmatrix} \mathbf{e}^{\top} = \begin{pmatrix} \hat{x}\check{y} \\ 0_{\mathbb{T}} \end{pmatrix},$$

for $\hat{x} \in \hat{\mathbb{G}}, \check{y}, \check{z} \in \check{\mathbb{H}}$ and $\mathbf{e} = (0, 1)$. Note that as $(\hat{x}a)\check{y} = \hat{x}(a\check{y})$ we will use the simpler notation $\hat{x}a\check{y} = (\hat{x}a)\check{y} = \hat{x}(a\check{y})$.

4.3.2 ElGamal encryption

We have already introduced the ElGamal encryption scheme in Section 2.4.1. In this section, we redefine it to leverage the more compact algebraic notation.

- **Setup**: on input a security parameter 1^λ, output a cyclic group $\hat{\mathbb{G}}$ of prime order p, an element $\hat{g} \in \hat{\mathbb{G}}$ and an element $\xi \leftarrow \mathbb{Z}_p^*$. Then, define the public key as $pk = (\hat{\mathbb{G}}, \hat{\mathbf{v}})$, where $\hat{\mathbf{v}} = (\xi\hat{g}, \hat{g})^\top \in \hat{\mathbb{G}}^{2\times 1}$ and the secret decryption key as $xk = (pk, \xi)$, where $\xi = (-\xi^{-1} \bmod p, 1)$.

- **Encrypt**: the encryption algorithm takes as input the setup gk, the public key pk and a message $\hat{x} \in \hat{\mathbb{G}}$, picks a random $r \leftarrow \mathbb{Z}_p$ and outputs the ciphertext $\hat{\mathbf{c}} = \mathbf{e}^\top \hat{x} + \hat{\mathbf{v}}r \in \hat{\mathbb{G}}^{2\times 1}$, where $\mathbf{e} = (0, 1)$.

- **Decrypt**: the decryption algorithm takes as input the setup gk, the secret key xk and a ciphertext $\hat{\mathbf{c}} \in \hat{\mathbb{G}}^{2\times 1}$ and outputs $\hat{x} = \xi\hat{\mathbf{c}}$. Note $\xi\mathbf{e}^\top = 1$ and $\xi\hat{\mathbf{v}} = 0$ so simple linear algebra shows decryption is correct.

As stated in Section 2.4.1, the ElGamal encryption scheme is IND-CPA secure if the DDH problem is computationally hard in $\hat{\mathbb{G}}$. ElGamal encryption can be defined similarly in $\check{\mathbb{H}}$ and if the SXDH assumption holds we then have IND-CPA secure encryption schemes in both $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$.

4.3.3 Pairing-product equations and other types of equations

Using the linear algebra friendly additive notation for group operations and multiplicative notation for the pairing, we can express the four types of equations for which Groth-Sahai allow us to make NIZK proofs in a compact way.

Consider elements $\hat{x}_1, \ldots, \hat{x}_m \in \hat{\mathbb{G}}$ and $\check{y}_1, \ldots, \check{y}_n \in \check{\mathbb{H}}$, which may be publicly known constants (called $\hat{a}_j$ and $\check{b}_i$ in the introduction) or secret variables. Let furthermore the matrix $\Gamma = \{\gamma_{ij}\}_{i=1,j=1}^{m,n} \in \mathbb{Z}_p^{m\times n}$ and $t_{\mathbb{T}} \in \mathbb{T}$ be public values. We can now write the pairing product equation simply as

$$\hat{\mathbf{x}}\Gamma\check{\mathbf{y}} = t_{\mathbb{T}},$$

where $\hat{\mathbf{x}} = (\hat{x}_1, \ldots, \hat{x}_m)$ and $\check{\mathbf{y}} = (\check{y}_1, \ldots, \check{y}_n)^\top$.

We can in a similar fashion write multi-scalar multiplication equations in $\hat{\mathbb{G}}$, multi-scalar multiplication equations in $\check{\mathbb{H}}$, and quadratic equations in $\mathbb{Z}_p$ as

$$\hat{\mathbf{x}}\Gamma\mathbf{y} = \hat{t} \qquad \mathbf{x}\Gamma\check{\mathbf{y}} = \check{t} \qquad \mathbf{x}\Gamma\mathbf{y} = t$$

for suitable choices of $m, n \in \mathbb{N}, \Gamma \in \mathbb{Z}_p^{m\times n}, \hat{\mathbf{x}} \in \hat{\mathbb{G}}^{1\times m}, \check{\mathbf{y}} \in \check{\mathbb{H}}^{n\times 1}, \mathbf{x} \in \mathbb{Z}_p^{1\times m}, \mathbf{y} \in \mathbb{Z}_p^{n\times 1}, \hat{t} \in \hat{\mathbb{G}}, \check{t} \in \check{\mathbb{H}}$ and $t \in \mathbb{Z}_p$. The vectors $\hat{\mathbf{x}}, \check{\mathbf{y}}, \mathbf{x}, \mathbf{y}$ may contain a mix of known public values and secret variables.

Groth and Sahai [GS12] made the useful observation that by subtracting $\hat{t} \cdot 1, 1 \cdot \check{t}$ and $1 \cdot t$ on both sides of the respective equations we may without loss of generality assume $\hat{t} = \hat{0}, \check{t} = \check{0}$ and $t = 0$ in all multi-scalar multiplication equations and quadratic equations. For multi-scalar multiplication equations this has an implicit cost, though: we need to treat the field element 1 as a variable, as the simulator will need to equivocate it to the field element 0. This means that if all the field elements that appear in the multi-scalar multiplication equation are constants, the equation will not be treated as a linear equation any more, which have witness-indistinguishable proofs which consist of 2 group elements. Instead, the zero-knowledge proof will consist of 6 group elements. However, in the special case where $\hat{t} = \hat{g}$ or $\check{t} = \check{h}$ we will show that we can still treat the equation as a linear equation, which have zero-knowledge proofs consisting of 2 group elements.

To get zero-knowledge proofs, we will in addition like Groth and Sahai restrict ourselves to $t_{\mathbb{T}} = 0_{\mathbb{T}}$ in all pairing product equations. Groth and Sahai [GS12] do not allow pairings of public constants in the pairing product equations in their zero-knowledge proofs, which we express by requiring the matrix Γ to contain entries $\gamma_{i,j} = 0$ whenever $\hat{x}_i$ and $\check{y}_j$ both are public values. This is because their zero-knowledge simulator breaks down when public values are paired. Groth and Sahai offers a work-around to deal with public values being paired with each other but it involves introducing additional multi-scalar multiplication equations and therefore increases the complexity of the zero-knowledge proof by many group elements. We will show that zero-knowledge simulation is possible when base elements $\hat{g}$ or $\check{h}$ are paired with each other or other public values. Since we do not need the additional multi-scalar multiplication equation used in Groth and Sahai's work-around this yields a significant efficiency gain whenever $\hat{g}$ or $\check{h}$ are paired with each other or other public values.

4.4 Commitment keys and commitments

As in the SXDH instantiation of [GS12], the setup algorithm Setup is defined as follows: given a security parameter 1^λ it generates a bilinear group $(p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathcal{PG}(1^\lambda)$. Also, like in Groth-Sahai proofs, commitment keys come in two flavours: extraction keys that give perfect soundness and simulation keys that give zero-knowledge. The two types of key generation algorithms are given in Fig. 4.1 and by the SXDH assumption extraction keys and simulation keys are computationally indistinguishable.[1]

The column vectors $\hat{\mathbf{v}}, \hat{\mathbf{w}}, \hat{\mathbf{u}} \in \hat{\mathbb{G}}^{2\times 1}$ will be used to make commitments $\hat{\mathbf{c}}$ to group elements $\hat{x} \in \hat{\mathbb{G}}$ and scalars $x \in \mathbb{Z}_p$. Commitments to group elements and scalars are computed as

$$\hat{\mathbf{c}} \leftarrow \mathbf{e}^\top \hat{x} + \hat{\mathbf{v}} r + \hat{\mathbf{w}} s \qquad \text{and} \qquad \hat{\mathbf{c}} \leftarrow \hat{\mathbf{u}} x + \hat{\mathbf{v}} r,$$

where $r, s \in \mathbb{Z}_p$. Commitments, usually denoted $\check{\mathbf{d}}$, to group elements $\hat{y} \in \check{\mathbb{H}}$ and scalars $y \in \mathbb{Z}_p$ are made analogously using the row vectors $\check{\mathbf{v}}, \check{\mathbf{w}}, \check{\mathbf{u}} \in \check{\mathbb{H}}^{1\times 2}$.

[1] The commitment keys are not defined exactly as in [GS12]: by defining $\hat{\mathbf{v}}$ as $(\xi\hat{g}, \hat{g})^\top$ instead of $(\hat{g}, \xi\hat{g})^\top$ we will be able to reduce the computational cost of the prover, as explained in Sec. 4.8. Besides this small difference, the keys $\hat{\mathbf{v}}, \hat{\mathbf{w}}, \hat{\mathbf{u}}, \check{\mathbf{v}}, \check{\mathbf{w}}$ and $\check{\mathbf{u}}$ correspond to u_1, u_2, u, v_1, v_2 and v in [GS12].

```
$\mathsf{ExtGenCRS}(gk)$
    $\rho \leftarrow \mathbb{Z}_p, \xi \leftarrow \mathbb{Z}_p^*$    $\sigma \leftarrow \mathbb{Z}_p, \psi \leftarrow \mathbb{Z}_p^*$
    $\hat{\mathbf{v}} \leftarrow (\xi\hat{g}, \hat{g})^\top$    $\check{\mathbf{v}} \leftarrow (\psi\check{h}, \check{h})$
    $\hat{\mathbf{w}} \leftarrow \rho\hat{\mathbf{v}}$    $\check{\mathbf{w}} \leftarrow \sigma\check{\mathbf{v}}$
    $\hat{\mathbf{u}} \leftarrow \hat{\mathbf{w}} + (\hat{0}, \hat{g})^\top$    $\check{\mathbf{u}} \leftarrow \check{\mathbf{w}} + (\check{0}, \check{h})$
    $\xi \leftarrow (-\xi^{-1} \bmod p, 1)$    $\psi \leftarrow (-\psi^{-1} \bmod p, 1)^\top$
$ck \leftarrow (p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{\mathbf{u}}, \hat{\mathbf{v}}, \hat{\mathbf{w}}, \check{\mathbf{u}}, \check{\mathbf{v}}, \check{\mathbf{w}})$
$xk \leftarrow (ck, \xi, \psi)$
Return $(ck, xk)$
```

```
$\mathsf{SimGenCRS}(gk)$
    $\rho \leftarrow \mathbb{Z}_p, \xi \leftarrow \mathbb{Z}_p^*$    $\sigma \leftarrow \mathbb{Z}_p, \psi \leftarrow \mathbb{Z}_p^*$
    $\hat{\mathbf{v}} \leftarrow (\xi\hat{g}, \hat{g})^\top$    $\check{\mathbf{v}} \leftarrow (\psi\check{h}, \check{h})$
    $\hat{\mathbf{w}} \leftarrow \rho\hat{\mathbf{v}} - (\hat{0}, \hat{g})^\top$    $\check{\mathbf{w}} \leftarrow \sigma\check{\mathbf{v}} - (\check{0}, \check{h})$
    $\hat{\mathbf{u}} \leftarrow \hat{\mathbf{w}} + (\hat{0}, \hat{g})^\top$    $\check{\mathbf{u}} \leftarrow \check{\mathbf{w}} + (\check{0}, \check{h})$
$ck \leftarrow (p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{\mathbf{u}}, \hat{\mathbf{v}}, \hat{\mathbf{w}}, \check{\mathbf{u}}, \check{\mathbf{v}}, \check{\mathbf{w}})$
$tk \leftarrow (ck, \rho, \sigma)$
Return $(ck, tk)$
```

Figure 4.1: Generator algorithms

The commitment scheme is similar to [GS12], however, we will have several different types of commitments and the randomness $r, s \in \mathbb{Z}_p$ we use will depend on the type. Fig. 4.2 summarizes the commitment types and describes the message, randomness and commitment spaces specified by the setup gk.

The type $t = (\text{pub}_{\hat{\mathbb{G}}}, \hat{x})$ corresponds to a commitment to a public value $\hat{x}$ using randomness $r = s = 0$. It is easy for the verifier to check whether a commitment $\hat{\mathbf{c}} = \mathbf{e}^\top\hat{x}$ is indeed a correct commitment to a public value $\hat{x}$. Explicitly allowing public values in the commitments simplifies the description of the proofs because we can now treat all elements $\hat{x}_1, \ldots, \hat{x}_m$ in a pairing product equation as committed values regardless of whether they are public or secret. Suppose some of the elements $\hat{\mathbf{x}} \in \hat{\mathbb{G}}^{1\times m}$ that appear in a pairing-product equation are committed as constant and others as Groth-Sahai commitments. The matrix consisting of all the commitments $\hat{C} = (\hat{\mathbf{c}}_1 \cdots \hat{\mathbf{c}}_m) \in \hat{\mathbb{G}}^{2\times m}$ can be written in a compact way as $\hat{C} = \mathbf{e}^\top\hat{\mathbf{x}} + \hat{\mathbf{v}}\mathbf{r}_x + \hat{\mathbf{w}}\mathbf{s}_x$, where for a constant $\hat{x}_i$ we just have $r_{x_i} = 0$ and $s_{x_i} = 0$.

In a standard Groth-Sahai proof, group element variables are committed as type $t = \text{com}_{\hat{\mathbb{G}}}$ using randomness $r, s \leftarrow \mathbb{Z}_p$. We will for greater efficiency also allow commitments of type $t = \text{enc}_{\hat{\mathbb{G}}}$ where $s = 0$. A type $t = \text{enc}_{\hat{\mathbb{G}}}$ commitment to a group element $\hat{x}$ is $\hat{\mathbf{c}} \leftarrow \mathbf{e}^\top\hat{x} + \hat{\mathbf{v}}r$, which is an ElGamal encryption of $\hat{x}$ as described in Sec. 4.3.2. Using encryption of variables instead of commitments reduces the computation and in some instances the size of the proofs. However, even on a simulation key the encryptions are only computationally hiding, so we must take care to ensure that it is possible to simulate proofs.

We also introduce the type $t = \text{base}_{\hat{\mathbb{G}}}$ for a commitment to the base element $\hat{g}$ using $r = s = 0$. This type allows us to differentiate $\hat{g}$ from other public values, which is important because the simulation becomes problematic when public values are paired with each other. However, in the special case when $\hat{g}$ is paired with $\check{h}$ or public constants it is possible to simulate. In addition, one can get shorter zero-knowledge proofs for certain equations by using the special properties of $\hat{g}$ and $\check{h}$.

t	m	(r, s)	$\hat{\mathbf{c}}$
$(\text{pub}_{\hat{\mathbb{G}}}, m)$	$\hat{m} \in \hat{\mathbb{G}}$	$r = s = 0$	$\hat{\mathbf{c}} = (\hat{0}, \hat{m})^\top$
$\text{enc}_{\hat{\mathbb{G}}}$	$\hat{m} \in \hat{\mathbb{G}}$	$r \in \mathbb{Z}_p, s = 0$	$\hat{\mathbf{c}} \in \hat{\mathbb{G}}^{2\times 1}$
$\text{com}_{\hat{\mathbb{G}}}$	$\hat{m} \in \hat{\mathbb{G}}$	$r, s \in \mathbb{Z}_p$	$\hat{\mathbf{c}} \in \hat{\mathbb{G}}^{2\times 1}$
$\text{base}_{\hat{\mathbb{G}}}$	$\hat{m} = \hat{g}$	$r = s = 0$	$\hat{\mathbf{c}} = (\hat{0}, \hat{g})^\top$
$\text{sca}_{\hat{\mathbb{G}}}$	$m \in \mathbb{Z}_p$	$r \in \mathbb{Z}_p, s = 0$	$\hat{\mathbf{c}} \in \hat{\mathbb{G}}^{2\times 1}$
$\text{unit}_{\hat{\mathbb{G}}}$	$m = 1$	$r = s = 0$	$\hat{\mathbf{c}} = \hat{\mathbf{u}}$

t	m	(r, s)	$\check{\mathbf{d}}$
$(\text{pub}_{\check{\mathbb{H}}}, m)$	$\check{m} \in \check{\mathbb{H}}$	$r = s = 0$	$\check{\mathbf{d}} = (\check{0}, \check{m})$
$\text{enc}_{\check{\mathbb{H}}}$	$\check{m} \in \check{\mathbb{H}}$	$r \in \mathbb{Z}_p, s = 0$	$\check{\mathbf{d}} \in \check{\mathbb{H}}^{1\times 2}$
$\text{com}_{\check{\mathbb{H}}}$	$\check{m} \in \check{\mathbb{H}}$	$r, s \in \mathbb{Z}_p$	$\check{\mathbf{d}} \in \check{\mathbb{H}}^{1\times 2}$
$\text{base}_{\check{\mathbb{H}}}$	$\check{m} = \check{h}$	$r = s = 0$	$\check{\mathbf{d}} = (\check{0}, \check{h})$
$\text{sca}_{\check{\mathbb{H}}}$	$m \in \mathbb{Z}_p$	$r \in \mathbb{Z}_p, s = 0$	$\check{\mathbf{d}} \in \check{\mathbb{H}}^{1\times 2}$
$\text{unit}_{\check{\mathbb{H}}}$	$m = 1$	$r = s = 0$	$\check{\mathbf{d}} = \check{\mathbf{u}}$

Figure 4.2: $\mathfrak{M}_{gk}, \mathfrak{R}_{gk}$ and $\mathfrak{C}_{gk}$.

Scalars have the type $t = \text{sca}_{\hat{\mathbb{G}}}$ and we use the type $t = \text{unit}_{\hat{\mathbb{G}}}$ for a commitment to the scalar 1 using $r = s = 0$. Please note that $t = \text{unit}_{\hat{\mathbb{G}}}$ suffices to incorporate any public value $a \in \mathbb{Z}_p$ into our equations by multiplying the corresponding row in the matrix Γ with a. With these two types, we can therefore commit to both variables and constants in $\mathbb{Z}_p$, which simplifies the description of the proofs.

We have now described the types of commitments in $\hat{\mathbb{G}}$ and similar types for commitments in $\check{\mathbb{H}}$ are given in Fig. 4.2. The commitment algorithm is described in Fig. 4.3.

The extraction key xk includes a vector ξ such that $\xi\hat{\mathbf{v}} = \xi\hat{\mathbf{w}} = \hat{0}$ and $\xi\mathbf{e}^\top = 1, \xi\hat{\mathbf{u}} = \hat{g}$. On a commitment to a group element $\hat{\mathbf{c}} = \mathbf{e}^\top\hat{x} + \hat{\mathbf{v}}r + \hat{\mathbf{w}}s$ or on an encryption to a group element $\hat{\mathbf{c}} = \mathbf{e}^\top\hat{x} + \hat{\mathbf{v}}r$ we can extract $\hat{x}$ by computing $\hat{x} = \xi\hat{\mathbf{c}}$. On a commitment to a scalar $\hat{\mathbf{c}} = \hat{\mathbf{u}}x + \hat{\mathbf{v}}r$ we extract $\hat{g}x = \xi\hat{\mathbf{c}}$, which uniquely determines the committed value x. The extraction algorithm is given in Fig. 4.4.

The simulated commitment algorithm $\mathsf{SimCommit}(gk, tk, t)$ is described in Fig. 4.5. Essentially it commits honestly to public constants, base elements $\hat{g}, \check{h}$ and units 1, which is easy to verify using public information. For all other types, it commits to 0.

On a simulation key, the commitments of types $\text{com}_{\hat{\mathbb{G}}}$ or $\text{sca}_{\hat{\mathbb{G}}}$ are perfectly hiding. Commitments of types $(\text{pub}_{\hat{\mathbb{G}}}, \hat{x})$ or $\text{enc}_{\hat{\mathbb{G}}}$ on the other hand are perfectly binding. However, by the SXDH assumption commitments of type $\text{enc}_{\hat{\mathbb{G}}}$ cannot be distinguished from commitments to other elements. Commitments of type $(\text{pub}_{\hat{\mathbb{G}}}, \hat{x})$ are public, so we do not require any hiding property.

Commitments to $\hat{g}$ and 1 of types $\text{base}_{\hat{\mathbb{G}}}$ and $\text{unit}_{\hat{\mathbb{G}}}$ are interesting. The secret simulation key specifies

Input	Randomness	Output
$(\text{pub}_{\hat{\mathbb{G}}}, \hat{x}), \hat{x}$	$r \leftarrow 0, s \leftarrow 0$	$\hat{\mathbf{c}} \leftarrow \mathbf{e}^\top \hat{x}$
$\text{enc}_{\hat{\mathbb{G}}}, \hat{x}$ $(\star)$	$r \leftarrow \mathbb{Z}_p, s \leftarrow 0$	$\hat{\mathbf{c}} \leftarrow \mathbf{e}^\top \hat{x} + \hat{\mathbf{v}} r$
$\text{com}_{\hat{\mathbb{G}}}, \hat{x}$	$r \leftarrow \mathbb{Z}_p, s \leftarrow \mathbb{Z}_p$	$\hat{\mathbf{c}} \leftarrow \mathbf{e}^\top \hat{x} + \hat{\mathbf{v}} r + \hat{\mathbf{w}} s$
$\text{base}_{\hat{\mathbb{G}}}, \hat{g}$ $(\star)$	$r \leftarrow 0, s \leftarrow 0$	$\hat{\mathbf{c}} \leftarrow \mathbf{e}^\top \hat{g}$
$\text{sca}_{\hat{\mathbb{G}}}, x$	$r \leftarrow \mathbb{Z}_p, s \leftarrow 0$	$\hat{\mathbf{c}} \leftarrow \hat{\mathbf{u}} x + \hat{\mathbf{v}} r$
$\text{unit}_{\hat{\mathbb{G}}}, 1$	$r \leftarrow 0, s \leftarrow 0$	$\hat{\mathbf{c}} \leftarrow \hat{\mathbf{u}}$

Input	Randomness	Output
$(\text{pub}_{\check{\mathbb{H}}}, \check{y}), \check{y}$	$r \leftarrow 0, s \leftarrow 0$	$\check{\mathbf{d}} \leftarrow \check{y} \mathbf{e}$
$\text{enc}_{\check{\mathbb{H}}}, \check{y}$ $(\star)$	$r \leftarrow \mathbb{Z}_p, s \leftarrow 0$	$\check{\mathbf{d}} \leftarrow \check{y} \mathbf{e} + r \check{\mathbf{v}}$
$\text{com}_{\check{\mathbb{H}}}, \check{y}$	$r \leftarrow \mathbb{Z}_p, s \leftarrow \mathbb{Z}_p$	$\check{\mathbf{d}} \leftarrow \check{y} \mathbf{e} + r \check{\mathbf{v}} + s \check{\mathbf{w}}$
$\text{base}_{\check{\mathbb{H}}}, \check{h}$ $(\star)$	$r \leftarrow 0, s \leftarrow 0$	$\check{\mathbf{d}} \leftarrow \check{h} \mathbf{e}$
$\text{sca}_{\check{\mathbb{H}}}, y$	$r \leftarrow \mathbb{Z}_p, s \leftarrow 0$	$\check{\mathbf{d}} \leftarrow y \check{\mathbf{u}} + r \check{\mathbf{v}}$
$\text{unit}_{\check{\mathbb{H}}}, 1$	$r \leftarrow 0, s \leftarrow 0$	$\check{\mathbf{d}} \leftarrow \check{\mathbf{u}}$

Figure 4.3: Commitment algorithm. The commitments marked with $(\star)$ were not defined in [GS12].

$\mathsf{Extract}(gk, xk, t, \hat{\mathbf{c}})$ where $\hat{\mathbf{c}} \in \hat{\mathbb{G}}^{2\times 1}$	$\mathsf{Extract}(gk, xk, t, \check{\mathbf{d}})$ where $\check{\mathbf{d}} \in \check{\mathbb{H}}^{1\times 2}$
Return $\hat{x} \leftarrow \xi \hat{\mathbf{c}}$	Return $\check{y} \leftarrow \check{\mathbf{d}} \psi$

Figure 4.4: Extraction algorithm.

Input	Randomness	Output	Input	Randomness	Output
$(\text{pub}_{\hat{\mathbb{G}}}, \hat{x})$	$r \leftarrow 0, s \leftarrow 0$	$\hat{\mathbf{c}} \leftarrow \mathbf{e}^\top \hat{x}$	$(\text{pub}_{\check{\mathbb{H}}}, \check{y})$	$r \leftarrow 0, s \leftarrow 0$	$\check{\mathbf{d}} \leftarrow \check{y} \mathbf{e}$
$\text{enc}_{\hat{\mathbb{G}}}$	$r \leftarrow \mathbb{Z}_p, s \leftarrow 0$	$\hat{\mathbf{c}} \leftarrow \hat{\mathbf{v}} r$	$\text{enc}_{\check{\mathbb{H}}}$	$r \leftarrow \mathbb{Z}_p, s \leftarrow 0$	$\check{\mathbf{d}} \leftarrow r \check{\mathbf{v}}$
$\text{com}_{\hat{\mathbb{G}}}$	$r \leftarrow \mathbb{Z}_p, s \leftarrow \mathbb{Z}_p$	$\hat{\mathbf{c}} \leftarrow \hat{\mathbf{v}} r + \hat{\mathbf{w}} s$	$\text{com}_{\check{\mathbb{H}}}$	$r \leftarrow \mathbb{Z}_p, s \leftarrow \mathbb{Z}_p$	$\check{\mathbf{d}} \leftarrow r \check{\mathbf{v}} + s \check{\mathbf{w}}$
$\text{base}_{\hat{\mathbb{G}}}$	$r \leftarrow 0, s \leftarrow 0$	$\hat{\mathbf{c}} \leftarrow \mathbf{e}^\top \hat{g}$	$\text{base}_{\check{\mathbb{H}}}$	$r \leftarrow 0, s \leftarrow 0$	$\check{\mathbf{d}} \leftarrow \check{h} \mathbf{e}$
$\text{sca}_{\hat{\mathbb{G}}}$	$r \leftarrow \mathbb{Z}_p, s \leftarrow 0$	$\hat{\mathbf{c}} \leftarrow \hat{\mathbf{v}} r$	$\text{sca}_{\check{\mathbb{H}}}$	$r \leftarrow \mathbb{Z}_p, s \leftarrow 0$	$\check{\mathbf{d}} \leftarrow r \check{\mathbf{v}}$
$\text{unit}_{\hat{\mathbb{G}}}$	$r \leftarrow 0, s \leftarrow 0$	$\hat{\mathbf{c}} \leftarrow \hat{\mathbf{u}}$	$\text{unit}_{\check{\mathbb{H}}}$	$r \leftarrow 0, s \leftarrow 0$	$\check{\mathbf{d}} \leftarrow \check{\mathbf{u}}$

Figure 4.5: Simulation algorithm for commitments.

ρ such that $\hat{\mathbf{u}} = \rho\hat{\mathbf{v}}$ and $\mathbf{e}^\top\hat{g} = \rho\hat{\mathbf{v}} - \hat{\mathbf{w}}$. This means that commitments of types $\mathsf{base}_{\hat{\mathbb{G}}}$ and $\mathsf{unit}_{\hat{\mathbb{G}}}$ can be equivocated as either commitments to $\hat{g}$ and 1 or as commitments to $\hat{0}$ and 0. The zero-knowledge simulator will use the equivocations to simulate proofs involving the base element $\hat{g}$ or constants in $\mathbb{Z}_p$.

4.5 Proofs

We will first explain how the proofs work using the example of pairing product equations to give intuition. We want to prove that committed values $\hat{\mathbf{x}}, \check{\mathbf{y}}$ satisfy the equation

$$\hat{\mathbf{x}}\Gamma\check{\mathbf{y}} = 0_{\mathbb{T}}.$$

Assume that we have committed to $\hat{\mathbf{x}}, \check{\mathbf{y}}$ as $\hat{C} = \mathbf{e}^\top\hat{\mathbf{x}} + \hat{\mathbf{v}}\mathbf{r}_x + \hat{\mathbf{w}}\mathbf{s}_x$ and $\check{D} = \check{\mathbf{y}}\mathbf{e} + \mathbf{r}_y\check{\mathbf{v}} + \mathbf{s}_y\check{\mathbf{w}}$. We then have

$$\begin{aligned}\hat{C}\Gamma\check{D} =&(\mathbf{e}^\top\hat{\mathbf{x}} + \hat{\mathbf{v}}\mathbf{r}_x + \hat{\mathbf{w}}\mathbf{s}_x)\Gamma(\check{\mathbf{y}}\mathbf{e} + \mathbf{r}_y\check{\mathbf{v}} + \mathbf{s}_y\check{\mathbf{w}})\\ =&\mathbf{e}^\top\hat{\mathbf{x}}\Gamma\check{\mathbf{y}}\mathbf{e} + \hat{\mathbf{v}}\mathbf{r}_x\Gamma\check{D} + \hat{\mathbf{w}}\mathbf{s}_x\Gamma\check{D} + \mathbf{e}^\top\hat{\mathbf{x}}\Gamma\mathbf{r}_y\check{\mathbf{v}} + \mathbf{e}^\top\hat{\mathbf{x}}\Gamma\mathbf{s}_y\check{\mathbf{w}}\\ =&0_{\mathbb{T}} + \hat{\mathbf{v}}\check{\pi}'_{\hat{v}} + \hat{\mathbf{w}}\check{\pi}'_{\hat{w}} + \hat{\pi}'_{\check{v}}\check{\mathbf{v}} + \hat{\pi}'_{\check{w}}\check{\mathbf{w}}\end{aligned}$$

where $\check{\pi}'_{\hat{v}} = \mathbf{r}_x\Gamma\check{D},\ \check{\pi}'_{\hat{w}} = \mathbf{s}_x\Gamma\check{D},\ \hat{\pi}'_{\check{v}} = \mathbf{e}^\top\hat{\mathbf{x}}\Gamma\mathbf{r}_y,\ \hat{\pi}'_{\check{w}} = \mathbf{e}^\top\hat{\mathbf{x}}\Gamma\mathbf{s}_y$.

The prover randomizes $\check{\pi}'_{\hat{v}}, \check{\pi}'_{\hat{w}}, \hat{\pi}'_{\check{v}}, \hat{\pi}'_{\check{w}}$ as $\check{\pi}_{\hat{v}} = \check{\pi}'_{\hat{v}} + \alpha\check{\mathbf{v}} + \beta\check{\mathbf{w}}$, $\check{\pi}_{\hat{w}} = \check{\pi}'_{\hat{w}} + \gamma\check{\mathbf{v}} + \delta\check{\mathbf{w}}$, $\hat{\pi}_{\check{v}} = \hat{\pi}'_{\check{v}} - \hat{\mathbf{v}}\alpha - \hat{\mathbf{w}}\gamma$, $\hat{\pi}_{\check{w}} = \hat{\pi}'_{\check{w}} - \hat{\mathbf{v}}\beta - \hat{\mathbf{w}}\delta$. This gives us a randomized proof $\check{\pi}_{\hat{v}}, \check{\pi}_{\hat{w}}, \hat{\pi}_{\check{v}}, \hat{\pi}_{\check{w}}$ satisfying the verification equation

$$\hat{C}\Gamma\check{D} = \hat{\mathbf{v}}\check{\pi}_{\hat{v}} + \hat{\mathbf{w}}\check{\pi}_{\hat{w}} + \hat{\pi}_{\check{v}}\check{\mathbf{v}} + \hat{\pi}_{\check{w}}\check{\mathbf{w}}.$$

Soundness and F-knowledge

An extraction key xk contains ξ, ψ such that $\xi\hat{\mathbf{v}} = \xi\hat{\mathbf{w}} = \hat{0}$ and $\check{\mathbf{v}}\psi = \check{\mathbf{w}}\psi = \check{0}$. Multiplying the verification equation by ξ and ψ on the left and right side respectively, we get

$$\xi\hat{C}\Gamma\check{D}\psi = \xi\hat{\mathbf{v}}\check{\pi}_{\hat{v}}\psi + \xi\hat{\mathbf{w}}\check{\pi}_{\hat{w}}\psi + \xi\hat{\pi}_{\check{v}}\check{\mathbf{v}}\psi + \xi\hat{\pi}_{\check{w}}\check{\mathbf{w}}\psi = 0_{\mathbb{T}}.$$

Observe, $\hat{\mathbf{x}} = \xi\hat{C}$ are the values the extractor $\mathsf{Extract}$ gets from the commitments $\hat{C}$ and $\check{\mathbf{y}} = \check{D}\psi$ are the values the extractor $\mathsf{Extract}$ gets from the commitments $\check{D}$. The extracted values from the commitments therefore satisfy $\hat{\mathbf{x}}\Gamma\check{\mathbf{y}} = 0_{\mathbb{T}}$. This gives us perfect soundness and perfect F-knowledge, where F on group elements in $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$ is the identity function.

Zero-knowledge

The simulator will simulate proofs by equivocating the commitments to values $\hat{\mathbf{x}}, \check{\mathbf{y}}$ that satisfy the equation $\hat{\mathbf{x}}\Gamma\check{\mathbf{y}} = 0_{\mathbb{T}}$. On a simulation key, commitments with types $\mathsf{com}_{\hat{\mathbb{G}}}, \mathsf{com}_{\check{\mathbb{H}}}$ are perfectly hiding. The simulator can therefore use $\hat{x}_i = \hat{0}$ or $\check{y}_j = \check{0}$. Commitments with types $\mathsf{base}_{\hat{\mathbb{G}}}, \mathsf{base}_{\check{\mathbb{H}}}$ are also equivocable to $\hat{0}$ or $\check{0}$

since on a simulation key $\mathbf{e}^\top \hat{g} = \hat{\mathbf{v}}\rho - \hat{\mathbf{w}}$ and $\check{h}\mathbf{e} = \sigma\check{\mathbf{v}} - \check{\mathbf{w}}$. By using equivocations to $\hat{0}$ and $\check{0}$ we can now ensure that $\hat{x}_i\gamma_{i,j}\check{y}_j = 0_{\mathbb{T}}$ whenever $t_{x_i} \in \{\text{base}_{\hat{\mathbb{G}}}, \text{com}_{\hat{\mathbb{G}}}\}$ or $t_{y_j} \in \{\text{base}_{\check{\mathbb{H}}}, \text{com}_{\check{\mathbb{H}}}\}$. Commitments of type $t_{x_i} \in \{(\text{pub}_{\hat{\mathbb{G}}}, \hat{x}), \text{enc}_{\hat{\mathbb{G}}}\}$ and $t_{y_j} \in \{(\text{pub}_{\check{\mathbb{H}}}, \check{y}), \text{enc}_{\check{\mathbb{H}}}\}$ cannot be equivocated and, to get zero-knowledge, we will therefore assume $\gamma_{i,j} = 0$ whenever such types are paired (as is also the case in [GS12]).

We now have that the simulator can equivocate commitments and base elements to $\hat{0}$ and $\check{0}$ such that the resulting $\hat{\mathbf{x}}, \check{\mathbf{y}}$ satisfy $\hat{\mathbf{x}}\Gamma\check{\mathbf{y}} = 0_{\mathbb{T}}$. The randomization of the proofs ensures that they will not leak information about whether we are giving a real proof or simulating. Recall the prover randomized $\check{\pi}'_{\hat{v}}, \check{\pi}'_{\hat{w}}, \hat{\pi}'_{\check{v}}, \hat{\pi}'_{\check{w}}$ as $\check{\pi}_{\hat{v}} = \check{\pi}'_{\hat{v}} + \alpha\check{\mathbf{v}} + \beta\check{\mathbf{w}}$, $\check{\pi}_{\hat{w}} = \check{\pi}'_{\hat{w}} + \gamma\check{\mathbf{v}} + \delta\check{\mathbf{w}}$, $\hat{\pi}_{\check{v}} = \hat{\pi}'_{\check{v}} - \hat{\mathbf{v}}\alpha - \hat{\mathbf{w}}\gamma$, $\hat{\pi}_{\check{w}} = \hat{\pi}'_{\check{w}} - \hat{\mathbf{v}}\beta - \hat{\mathbf{w}}\delta$. On a simulation key this means regardless of whether we are giving a real proof or a simulated proof $\check{\pi}_{\hat{v}}, \check{\pi}_{\hat{w}}$ are uniformly random and $\hat{\pi}_{\check{v}}, \hat{\pi}_{\check{w}}$ are the unique values that make the verification equation true. Finally, the encrypted elements are computationally hidden by the SXDH assumption, so here the simulator may use encryptions of $\hat{0}$ and $\check{0}$ instead of the witness and as we shall show the proofs can be constructed on top of the ciphertexts such that they do not reveal whether the underlying plaintext are part of a real witness or are set to zero by the simulator.

Optimizations

Now let us return to the prover. Observe that $\mathbf{r}_x, \mathbf{s}_x, \mathbf{r}_y, \mathbf{s}_y$ may have some zero elements. In particular, assume that all elements in $\mathbf{s}_x$ are 0. This happens if all $\hat{x}_i$ in the statement have types $\text{enc}_{\hat{\mathbb{G}}}$, $(\text{pub}_{\hat{\mathbb{G}}}, \hat{x}_i)$ or $\text{base}_{\hat{\mathbb{G}}}$. Moreover, assume that all elements $\check{\mathbf{y}}$ have as types either $\text{com}_{\check{\mathbb{H}}}$ or $\text{base}_{\check{\mathbb{H}}}$ so that a simulator uses $\check{\mathbf{y}} = \check{\mathbf{0}}$ in the simulated proof. This sets $\check{\pi}'_{\hat{w}} = \check{0}$. As $\check{\pi}'_{\hat{w}}$ is the same for all witnesses, even for "simulated witnesses", we might as well set $\gamma = \delta = 0$. This gives us a proof that consists of 4 elements in $\hat{\mathbb{G}}$ and 2 elements in $\check{\mathbb{H}}$ instead of 4 elements both in $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$. For such equations, we therefore save 2 group elements or 25% of the proof size compared to Groth and Sahai [GS12] where there is no $\text{enc}_{\hat{\mathbb{G}}}$ or $\text{enc}_{\check{\mathbb{H}}}$ types. We refer to Fig. 4.9 for the list of equation types and the corresponding proof sizes.

4.5.1 The full proof system

We divide the possible statements into 16 different types. They are summarized in Fig. 4.6, which provides an algorithm for checking that the statement format is correct.

The relation R is defined in Fig. 4.7, which provides an algorithm to check whether a statement is true. The relation first checks that the types of the witnesses and the types of the equations match according to Fig. 4.6 and then whether the relevant pairing product, multi-scalar multiplication or quadratic equation is satisfied.

The prover and verifier are given in Fig. 4.8. The prover constructs a proof for the relevant type of equation assuming the input is a correctly formatted statement with valid openings of commitments to a satisfying witness. The verifier uses the matching verification equation to check validity of a proof.

CheckFormat$(gk, T, \Gamma, \{t_{x_i}\}_{i=1}^{m}, \{t_{y_j}\}_{j=1}^{n})$

Check $\Gamma \in \mathbb{Z}_p^{m \times n}$

Check that the equation and message types match each other according to the table below

T	$t_{x_1}, \ldots, t_{x_m}$	$t_{y_1}, \ldots, t_{y_n}$
PPE	$\text{base}_{\hat{\mathbb{G}}}, (\text{pub}_{\hat{\mathbb{G}}}, \hat{x}_i), \text{enc}_{\hat{\mathbb{G}}}, \text{com}_{\hat{\mathbb{G}}}$	$\text{base}_{\check{\mathbb{H}}}, (\text{pub}_{\check{\mathbb{H}}}, \check{y}_j), \text{enc}_{\check{\mathbb{H}}}, \text{com}_{\check{\mathbb{H}}}$
$\text{PEnc}_{\hat{\mathbb{G}}}$	$\text{base}_{\hat{\mathbb{G}}}, (\text{pub}_{\hat{\mathbb{G}}}, \hat{x}_i), \text{enc}_{\hat{\mathbb{G}}}$	$\text{base}_{\check{\mathbb{H}}}, \text{com}_{\check{\mathbb{H}}}$
$\text{PConst}_{\hat{\mathbb{G}}}$	$\text{base}_{\hat{\mathbb{G}}}, (\text{pub}_{\hat{\mathbb{G}}}, \hat{x}_i)$	$\text{base}_{\check{\mathbb{H}}}, \text{com}_{\check{\mathbb{H}}}$
$\text{PEnc}_{\check{\mathbb{H}}}$	$\text{base}_{\hat{\mathbb{G}}}, \text{com}_{\hat{\mathbb{G}}}$	$\text{base}_{\check{\mathbb{H}}}, (\text{pub}_{\check{\mathbb{H}}}, \check{y}_j), \text{enc}_{\check{\mathbb{H}}}$
$\text{PConst}_{\check{\mathbb{H}}}$	$\text{base}_{\hat{\mathbb{G}}}, \text{com}_{\hat{\mathbb{G}}}$	$\text{base}_{\check{\mathbb{H}}}, (\text{pub}_{\check{\mathbb{H}}}, \check{y}_j)$
$\text{ME}_{\hat{\mathbb{G}}}$	$\text{base}_{\hat{\mathbb{G}}}, (\text{pub}_{\hat{\mathbb{G}}}, \hat{x}_i), \text{enc}_{\hat{\mathbb{G}}}, \text{com}_{\hat{\mathbb{G}}}$	$\text{unit}_{\check{\mathbb{H}}}, \text{sca}_{\check{\mathbb{H}}}$
$\text{MEnc}_{\hat{\mathbb{G}}}$	$\text{base}_{\hat{\mathbb{G}}}, (\text{pub}_{\hat{\mathbb{G}}}, \hat{x}_i), \text{enc}_{\hat{\mathbb{G}}}$	$\text{unit}_{\check{\mathbb{H}}}, \text{sca}_{\check{\mathbb{H}}}$
$\text{MConst}_{\hat{\mathbb{G}}}$	$\text{base}_{\hat{\mathbb{G}}}, (\text{pub}_{\hat{\mathbb{G}}}, \hat{x}_i)$	$\text{unit}_{\check{\mathbb{H}}}, \text{sca}_{\check{\mathbb{H}}}$
$\text{MLin}_{\hat{\mathbb{G}}}$	$\text{base}_{\hat{\mathbb{G}}}, \text{com}_{\hat{\mathbb{G}}}$	$\text{unit}_{\check{\mathbb{H}}}$
$\text{ME}_{\check{\mathbb{H}}}$	$\text{unit}_{\hat{\mathbb{G}}}, \text{sca}_{\hat{\mathbb{G}}}$	$\text{base}_{\check{\mathbb{H}}}, (\text{pub}_{\check{\mathbb{H}}}, \check{y}_j), \text{enc}_{\check{\mathbb{H}}}, \text{com}_{\check{\mathbb{H}}}$
$\text{MEnc}_{\check{\mathbb{H}}}$	$\text{unit}_{\hat{\mathbb{G}}}, \text{sca}_{\hat{\mathbb{G}}}$	$\text{base}_{\check{\mathbb{H}}}, (\text{pub}_{\check{\mathbb{H}}}, \check{y}_j), \text{enc}_{\check{\mathbb{H}}}$
$\text{MConst}_{\check{\mathbb{H}}}$	$\text{unit}_{\hat{\mathbb{G}}}, \text{sca}_{\hat{\mathbb{G}}}$	$\text{base}_{\check{\mathbb{H}}}, (\text{pub}_{\check{\mathbb{H}}}, \check{y}_j)$
$\text{MLin}_{\check{\mathbb{H}}}$	$\text{unit}_{\hat{\mathbb{G}}}$	$\text{base}_{\check{\mathbb{H}}}, \text{com}_{\check{\mathbb{H}}}$
QE	$\text{unit}_{\hat{\mathbb{G}}}, \text{sca}_{\hat{\mathbb{G}}}$	$\text{unit}_{\check{\mathbb{H}}}, \text{sca}_{\check{\mathbb{H}}}$
$\text{QConst}_{\hat{\mathbb{G}}}$	$\text{unit}_{\hat{\mathbb{G}}}$	$\text{unit}_{\check{\mathbb{H}}}, \text{sca}_{\check{\mathbb{H}}}$
$\text{QConst}_{\check{\mathbb{H}}}$	$\text{unit}_{\hat{\mathbb{G}}}, \text{sca}_{\hat{\mathbb{G}}}$	$\text{unit}_{\check{\mathbb{H}}}$

If $T = \text{PPE}$ check $\Gamma_{i,j} = 0$ for all (i, j) where $t_{x_i} \in \{(\text{pub}_{\hat{\mathbb{G}}}, \hat{x}_i), \text{enc}_{\hat{\mathbb{G}}}\}$ and $t_{y_j} \in \{(\text{pub}_{\check{\mathbb{H}}}, \check{y}_j), \text{enc}_{\check{\mathbb{H}}}\}$

Accept format if all checks pass, else abort

Figure 4.6: Equation - message types check

```
$R(gk, (T, \Gamma), (\{(t_{x_i}, x_i)\}_{i=1}^m, \{(t_{y_j}, y_j)\}_{j=1}^n))$
CheckFormat$(gk, T, \Gamma, \{t_{x_i}\}_{i=1}^m, \{t_{y_j}\}_{j=1}^n)$
For all $i, j$ check $(t_{x_i}, x_i) \in \mathfrak{M}_{gk}$ and $(t_{y_j}, y_j) \in \mathfrak{M}_{gk}$
If $\mathbf{x} \in \hat{\mathbb{G}}^m$ and $\mathbf{y} \in \check{\mathbb{H}}^n$ check $\mathbf{x}\Gamma\mathbf{y} = 0_{\mathbb{T}}$
If $\mathbf{x} \in \hat{\mathbb{G}}^m$ and $\mathbf{y} \in \mathbb{Z}_p^n$ check $\mathbf{x}\Gamma\mathbf{y} = \hat{0}$
If $\mathbf{x} \in \mathbb{Z}_p^m$ and $\mathbf{y} \in \check{\mathbb{H}}^n$ check $\mathbf{x}\Gamma\mathbf{y} = \check{0}$
If $\mathbf{x} \in \mathbb{Z}_p^m$ and $\mathbf{y} \in \mathbb{Z}_p^n$ check $\mathbf{x}\Gamma\mathbf{y} = 0$
Accept if and only if all checks pass
```

Figure 4.7: Relation that defines the setup-dependent languages for our proofs

```
Prove$(gk, ck, T, \Gamma, \{(t_{x_i}, x_i, (r_{x_i}, s_{x_i}))\}_{i=1}^m, \{(t_{y_j}, y_j, (r_{y_j}, s_{y_j}))\}_{j=1}^n)$
If $\mathbf{x} \in \hat{\mathbb{G}}^m$ define $\hat{C} = \mathbf{e}^\top \mathbf{x} + \hat{\mathbf{v}}\mathbf{r}_x + \hat{\mathbf{w}}\mathbf{s}_x$ else if $\mathbf{x} \in \mathbb{Z}_p^m$ define $\hat{C} = \hat{\mathbf{u}}\mathbf{x} + \hat{\mathbf{v}}\mathbf{r}_x$
If $\mathbf{y} \in \check{\mathbb{H}}^n$ define $\check{D} = \mathbf{y}\mathbf{e} + \mathbf{r}_y\check{\mathbf{v}} + \mathbf{s}_y\check{\mathbf{w}}$ else if $\mathbf{y} \in \mathbb{Z}_p^n$ define $\check{D} = \mathbf{y}\check{\mathbf{u}} + \mathbf{r}_y\check{\mathbf{v}}$
Set $\alpha = \beta = \gamma = \delta = 0$
If $T =$ PPE pick $\alpha, \beta, \gamma, \delta \leftarrow \mathbb{Z}_p$
If $T \in \{\mathrm{PEnc}_{\hat{\mathbb{G}}}, \mathrm{ME}_{\check{\mathbb{H}}}\}$ pick $\alpha, \beta \leftarrow \mathbb{Z}_p$
If $T \in \{\mathrm{PEnc}_{\check{\mathbb{H}}}, \mathrm{ME}_{\hat{\mathbb{G}}}\}$ pick $\alpha, \gamma \leftarrow \mathbb{Z}_p$
If $T \in \{\mathrm{MEnc}_{\hat{\mathbb{G}}}, \mathrm{MEnc}_{\check{\mathbb{H}}}, \mathrm{QE}\}$ pick $\alpha \leftarrow \mathbb{Z}_p$
    $\check{\pi}_{\hat{v}} \leftarrow \mathbf{r}_x\Gamma\check{D} + \alpha\check{\mathbf{v}} + \beta\check{\mathbf{w}}$        $\hat{\pi}_{\check{v}} \leftarrow (\hat{C} - \hat{\mathbf{v}}\mathbf{r}_x - \hat{\mathbf{w}}\mathbf{s}_x)\Gamma\mathbf{r}_y - \hat{\mathbf{v}}\alpha - \hat{\mathbf{w}}\gamma$
    $\check{\pi}_{\hat{w}} \leftarrow \mathbf{s}_x\Gamma\check{D} + \gamma\check{\mathbf{v}} + \delta\check{\mathbf{w}}$        $\hat{\pi}_{\check{w}} \leftarrow (\hat{C} - \hat{\mathbf{v}}\mathbf{r}_x - \hat{\mathbf{w}}\mathbf{s}_x)\Gamma\mathbf{s}_y - \hat{\mathbf{v}}\beta - \hat{\mathbf{w}}\delta$
Return $\pi = (\check{\pi}_{\hat{v}}, \check{\pi}_{\hat{w}}, \hat{\pi}_{\check{v}}, \hat{\pi}_{\check{w}})$
```

```
VerifyProof$(gk, ck, T, \Gamma, \{(t_{x_i}, \hat{\mathbf{c}}_i)\}_{i=1}^m, \{(t_{y_j}, \check{\mathbf{d}}_j)\}_{j=1}^n, \pi)$
CheckFormat$_{ck}(T, \Gamma, \{t_{x_i}\}_{i=1}^m, \{t_{y_j}\}_{j=1}^n)$
Check $\hat{C} = (\hat{\mathbf{c}}_1 \cdots \hat{\mathbf{c}}_m) \in \hat{\mathbb{G}}^{2\times m}$ and $\check{D} = \left(\check{\mathbf{d}}_1 \cdots \check{\mathbf{d}}_n\right)^\top \in \check{\mathbb{H}}^{n\times 2}$
Check $\pi = (\check{\pi}_{\hat{v}}, \check{\pi}_{\hat{w}}, \hat{\pi}_{\check{v}}, \hat{\pi}_{\check{w}}) \in \check{\mathbb{H}}^{2\times 1} \times \check{\mathbb{H}}^{2\times 1} \times \hat{\mathbb{G}}^{1\times 2} \times \hat{\mathbb{G}}^{1\times 2}$
Check $\hat{C}\Gamma\check{D} = \hat{\mathbf{v}}\check{\pi}_{\hat{v}} + \hat{\mathbf{w}}\check{\pi}_{\hat{w}} + \hat{\pi}_{\check{v}}\check{\mathbf{v}} + \hat{\pi}_{\check{w}}\check{\mathbf{w}}$
Return 1 if all checks pass, else return 0
```

Figure 4.8: Prover and verifier algorithms

T	$\check{\pi}_{\hat{v}}$	$\check{\pi}_{\hat{w}}$	$\hat{\pi}_{\check{v}}$	$\hat{\pi}_{\check{w}}$	$\hat{\mathbb{G}}$	$\check{\mathbb{H}}$	$\mathbb{Z}_p$
PPE	$\check{\pi}_{\hat{v}}$	$\check{\pi}_{\hat{w}}$	$\hat{\pi}_{\check{v}}$	$\hat{\pi}_{\check{w}}$	4	4	0
PEnc$_{\hat{\mathbb{G}}}$ $(\star)$	$\check{\pi}_{\hat{v}}$	$\check{\mathbf{0}}$	$\hat{\pi}_{\check{v}}$	$\hat{\pi}_{\check{w}}$	4	2	0
PConst$_{\hat{\mathbb{G}}}$	$\check{\mathbf{0}}$	$\check{\mathbf{0}}$	$\mathbf{e}^\top(\hat{\mathbf{x}}\Gamma\mathbf{r}_y)$	$\mathbf{e}^\top(\hat{\mathbf{x}}\Gamma\mathbf{s}_y)$	2	0	0
PEnc$_{\check{\mathbb{H}}}$ $(\star)$	$\check{\pi}_{\hat{v}}$	$\check{\pi}_{\hat{w}}$	$\hat{\pi}_{\check{v}}$	$\hat{\mathbf{0}}$	2	4	0
PConst$_{\check{\mathbb{H}}}$	$(\mathbf{r}_x\Gamma\check{\mathbf{y}})\mathbf{e}$	$(\mathbf{s}_x\Gamma\check{\mathbf{y}})\mathbf{e}$	$\hat{\mathbf{0}}$	$\hat{\mathbf{0}}$	0	2	0
ME$_{\hat{\mathbb{G}}}$	$\check{\pi}_{\hat{v}}$	$\check{\pi}_{\hat{w}}$	$\hat{\pi}_{\check{v}}$	$\hat{\mathbf{0}}$	2	4	0
MEnc$_{\hat{\mathbb{G}}}$ $(\star)$	$\check{\pi}_{\hat{v}}$	$\hat{\mathbf{0}}$	$\hat{\pi}_{\check{v}}$	$\hat{\mathbf{0}}$	2	2	0
MConst$_{\hat{\mathbb{G}}}$	$\check{\mathbf{0}}$	$\check{\mathbf{0}}$	$\mathbf{e}^\top(\hat{\mathbf{x}}\Gamma\mathbf{r}_y)$	$\hat{\mathbf{0}}$	1	0	0
MLin$_{\hat{\mathbb{G}}}$	$(\mathbf{r}_x\Gamma\mathbf{y})\check{\mathbf{u}}$	$(\mathbf{s}_x\Gamma\mathbf{y})\check{\mathbf{u}}$	$\hat{\mathbf{0}}$	$\hat{\mathbf{0}}$	0	0	2
ME$_{\check{\mathbb{H}}}$	$\check{\pi}_{\hat{v}}$	$\check{\mathbf{0}}$	$\hat{\pi}_{\check{v}}$	$\hat{\pi}_{\check{w}}$	4	2	0
MEnc$_{\check{\mathbb{H}}}$ $(\star)$	$\check{\pi}_{\hat{v}}$	$\check{\mathbf{0}}$	$\hat{\pi}_{\check{v}}$	$\hat{\mathbf{0}}$	2	2	0
MConst$_{\check{\mathbb{H}}}$	$(\mathbf{r}_x\Gamma\check{\mathbf{y}})\mathbf{e}$	$\check{\mathbf{0}}$	$\hat{\mathbf{0}}$	$\hat{\mathbf{0}}$	0	1	0
MLin$_{\check{\mathbb{H}}}$	$\check{\mathbf{0}}$	$\check{\mathbf{0}}$	$\hat{\mathbf{u}}(\mathbf{x}\Gamma\mathbf{r}_y)$	$\hat{\mathbf{u}}(\mathbf{x}\Gamma\mathbf{s}_y)$	0	0	2
QE	$\check{\pi}_{\hat{v}}$	$\check{\mathbf{0}}$	$\hat{\pi}_{\check{v}}$	$\hat{\mathbf{0}}$	2	2	0
QConst$_{\hat{\mathbb{G}}}$	$\check{\mathbf{0}}$	$\check{\mathbf{0}}$	$\hat{\mathbf{u}}(\mathbf{x}\Gamma\mathbf{r}_y)$	$\hat{\mathbf{0}}$	0	0	1
QConst$_{\check{\mathbb{H}}}$	$(\mathbf{r}_x\Gamma\mathbf{y})\check{\mathbf{u}}$	$\check{\mathbf{0}}$	$\hat{\mathbf{0}}$	$\hat{\mathbf{0}}$	0	0	1

Figure 4.9: Size of an NIZK proof for each type of equation. The size of the NIZK proofs are the same as in [GS12], except for the equation types marked with $(\star)$, which were not defined in [GS12].

As described earlier there are some types of equations where parts of the proof are just 0 or can be compressed. In Fig. 4.9 we show the parts of the proofs that allow reduced communication for each type of equation and, if the proof is shortened, we explain how it is done.

Let F be given by

$$\begin{array}{ll} F(gk,t,\hat{x}) = \hat{x} & \text{for } t \in \{(\text{pub}_{\hat{\mathbb{G}}},\hat{x}), \text{enc}_{\hat{\mathbb{G}}}, \text{com}_{\hat{\mathbb{G}}}, \text{base}_{\hat{\mathbb{G}}}\} \\ F(gk,t,x) = \hat{g}x & \text{for } t \in \{\text{sca}_{\hat{\mathbb{G}}}, \text{unit}_{\hat{\mathbb{G}}}\} \\ F(gk,t,\hat{y}) = \check{y} & \text{for } t \in \{(\text{pub}_{\check{\mathbb{H}}},\check{y}), \text{enc}_{\check{\mathbb{H}}}, \text{com}_{\check{\mathbb{H}}}, \text{base}_{\check{\mathbb{H}}}\} \\ F(gk,t,y) = y\check{h} & \text{for } t \in \{\text{sca}_{\check{\mathbb{H}}}, \text{unit}_{\check{\mathbb{H}}}\} \end{array} .$$

Theorem 4.5.1. *The commit-and-prove scheme given in Figs. 4.1,4.3,4.4 and 4.8 has perfect correctness, perfect soundness and F-knowledge for the function F defined above, and computational composable zero-knowledge if the SXDH assumption holds relative to $\mathcal{PG}$.*

The description of the zero-knowledge simulator is given in Sec. 4.6. The proof of the theorem follows from Lemmas 4.7.1, 4.7.2 and 4.7.3 given in Sec. 4.7.

$\mathsf{SimProve}(gk, tk, T, \Gamma, \{(t_{x_i}, (x_i, r_{x_i}, s_{x_i}))\}_{i=1}^m, \{(t_{y_j}, (y_j, r_{y_j}, s_{y_j}))\}_{j=1}^n)$

Modify some values according to the following tables

T	t_{x_i}	x_i, r_{x_i}, s_{x_i}
PPE, $\text{PEnc}_{\check{\mathbb{H}}}$, $\text{PConst}_{\check{\mathbb{H}}}$, $\text{MLin}_{\hat{\mathbb{G}}}$	$\text{base}_{\hat{\mathbb{G}}}$	$x_i \leftarrow \hat{0}, r_{x_i} \leftarrow \rho, s_{x_i} \leftarrow -1$
$\text{ME}_{\check{\mathbb{H}}}$, $\text{MEnc}_{\check{\mathbb{H}}}$, $\text{MConst}_{\check{\mathbb{H}}}$, QE, $\text{QConst}_{\check{\mathbb{H}}}$	$\text{unit}_{\hat{\mathbb{G}}}$	$x_i \leftarrow 0, r_{x_i} \leftarrow \rho, s_{x_i} \leftarrow 0$

T	t_{y_j}	y_j, r_{y_j}, s_{y_j}
PPE, $\text{PEnc}_{\hat{\mathbb{G}}}$, $\text{PConst}_{\hat{\mathbb{G}}}$, $\text{MLin}_{\check{\mathbb{H}}}$	$\text{base}_{\check{\mathbb{H}}}$	$y_j \leftarrow \check{0}, r_{y_j} \leftarrow \sigma, s_{y_j} \leftarrow -1$
$\text{ME}_{\hat{\mathbb{G}}}$, $\text{MEnc}_{\hat{\mathbb{G}}}$, $\text{MConst}_{\hat{\mathbb{G}}}$, QE, $\text{QConst}_{\hat{\mathbb{G}}}$	$\text{unit}_{\check{\mathbb{H}}}$	$y_j \leftarrow 0, r_{y_j} \leftarrow \sigma, s_{y_j} \leftarrow 0$

If $\mathbf{x} \in \hat{\mathbb{G}}^m$ define $\hat{C} = \mathbf{e}^\top \mathbf{x} + \hat{\mathbf{v}}\mathbf{r}_x + \hat{\mathbf{w}}\mathbf{s}_x$ else if $\mathbf{x} \in \mathbb{Z}_p^m$ define $\hat{C} = \hat{\mathbf{u}}\mathbf{x} + \hat{\mathbf{v}}\mathbf{r}_x$

If $\mathbf{y} \in \check{\mathbb{H}}^n$ define $\check{D} = \mathbf{y}\mathbf{e} + \mathbf{r}_y\check{\mathbf{v}} + \mathbf{s}_y\check{\mathbf{w}}$ else if $\mathbf{y} \in \mathbb{Z}_p^n$ define $\check{D} = \mathbf{y}\check{\mathbf{u}} + \mathbf{r}_y\check{\mathbf{v}}$

Set $\alpha = \beta = \gamma = \delta = 0$

If $T =$ PPE pick $\alpha, \beta, \gamma, \delta \leftarrow \mathbb{Z}_p$

If $T \in \{\text{PEnc}_{\hat{\mathbb{G}}}, \text{ME}_{\check{\mathbb{H}}}\}$ pick $\alpha, \beta \leftarrow \mathbb{Z}_p$

If $T \in \{\text{PEnc}_{\check{\mathbb{H}}}, \text{ME}_{\hat{\mathbb{G}}}\}$ pick $\alpha, \gamma \leftarrow \mathbb{Z}_p$

If $T \in \{\text{MEnc}_{\hat{\mathbb{G}}}, \text{MEnc}_{\check{\mathbb{H}}}, \text{QE}\}$ pick $\alpha \leftarrow \mathbb{Z}_p$

$\check{\pi}_{\hat{v}} \leftarrow \mathbf{r}_x\Gamma\check{D} + \alpha\check{\mathbf{v}} + \beta\check{\mathbf{w}}$ $\qquad$ $\hat{\pi}_{\check{v}} \leftarrow (\hat{C} - \hat{\mathbf{v}}\mathbf{r}_x - \hat{\mathbf{w}}\mathbf{s}_x)\Gamma\mathbf{r}_y - \hat{\mathbf{v}}\alpha - \hat{\mathbf{w}}\gamma$

$\check{\pi}_{\hat{w}} \leftarrow \mathbf{s}_x\Gamma\check{D} + \gamma\check{\mathbf{v}} + \delta\check{\mathbf{w}}$ $\qquad$ $\hat{\pi}_{\check{w}} \leftarrow (\hat{C} - \hat{\mathbf{v}}\mathbf{r}_x - \hat{\mathbf{w}}\mathbf{s}_x)\Gamma\mathbf{s}_y - \hat{\mathbf{v}}\beta - \hat{\mathbf{w}}\delta$

Return $\pi = (\check{\pi}_{\hat{v}}, \check{\pi}_{\hat{w}}, \hat{\pi}_{\check{v}}, \hat{\pi}_{\check{w}})$

Figure 4.10: Simulator algorithm

4.6 Zero-Knowledge simulator

Simulation

The simulator gets as input a well-formed statement where the equation and message types match. Unlike the prover, the simulator does not know the witness. Some of the simulator's inputs x_i and y_j will correspond to public values, base elements and units, while others have been generated with the simulated commitments and are 0. Having the secret simulation key the simulator can equivocate the $\text{base}_{\hat{\mathbb{G}}}$, $\text{unit}_{\hat{\mathbb{G}}}$ and $\text{base}_{\check{\mathbb{H}}}$, $\text{unit}_{\check{\mathbb{H}}}$ type commitments to 0. This enables it to get a satisfying witness for the statement and using this simulated witness it can now create the proof as an honest prover would do. The simulator algorithm is detailed in Fig. 4.10.

4.7 Security proofs for the commit-and-prove scheme

Lemma 4.7.1. *The commit-and-prove scheme has perfect correctness.*

Proof. The prover gets a statement and valid witness $(gk, (T, \Gamma), \{(t_{x_i}, x_i)\}_{i=1}^m, \{(t_{y_n}, y_n)\}_{j=1}^n) \in R$, which guarantees the statement is correctly formatted, $(t_{x_i}, x_i), (t_{y_j}, y_j) \in \mathfrak{M}_{gk}$ and the vectors $\mathbf{x} = (x_1, \dots, x_m)$ and $\mathbf{y} = (y_1, \dots, y_n)^\top$ fall into one out of four cases given below.

Case	Implication
$\mathbf{x} \in \hat{\mathbb{G}}^{1\times m}, \mathbf{y} \in \check{\mathbb{H}}^{n\times 1}$ and $\mathbf{x}\Gamma\mathbf{y} = 0_{\mathbb{T}}$	$\mathbf{e}^\top\mathbf{x}\Gamma\mathbf{y}\mathbf{e} = 0_{\mathbb{T}}$
$\mathbf{x} \in \hat{\mathbb{G}}^{1\times m}, \mathbf{y} \in \mathbb{Z}_p^{n\times 1}$ and $\mathbf{x}\Gamma\mathbf{y} = \hat{0}$	$\mathbf{e}^\top\mathbf{x}\Gamma\mathbf{y}\check{\mathbf{u}} = 0_{\mathbb{T}}$
$\mathbf{x} \in \mathbb{Z}_p^{1\times m}, \mathbf{y} \in \check{\mathbb{H}}^{n\times 1}$ and $\mathbf{x}\Gamma\mathbf{y} = \check{0}$	$\hat{\mathbf{u}}\mathbf{x}\Gamma\mathbf{y}\mathbf{e} = 0_{\mathbb{T}}$
$\mathbf{x} \in \mathbb{Z}_p^{1\times m}, \mathbf{y} \in \mathbb{Z}_p^{n\times 1}$ and $\mathbf{x}\Gamma\mathbf{y} = 0$	$\hat{\mathbf{u}}\mathbf{x}\Gamma\mathbf{y}\check{\mathbf{u}} = 0_{\mathbb{T}}$

The prover's input also satisfies $(t_{x_i}, (r_{x_i}, s_{x_i})), (t_{y_j}, (r_{y_j}, s_{y_j})) \in \mathfrak{R}_{gk}$. For $\mathbf{x} \in \mathbb{Z}_p^m$ this gives us $\mathbf{s}_x = \mathbf{0}$ and we can write $\hat{C} = \hat{\mathbf{u}}\mathbf{x} + \hat{\mathbf{v}}\mathbf{r}_x + \hat{\mathbf{w}}\mathbf{s}_x$. Similarly, for $\mathbf{y} \in \mathbb{Z}_p^n$ it gives us $\mathbf{s}_y = \mathbf{0}$ and we can write $\check{D} = \mathbf{y}\check{\mathbf{u}} + \mathbf{r}_y\check{\mathbf{v}} + \mathbf{s}_y\check{\mathbf{w}}$.

This means that for each of the four cases

$$(\hat{C} - \hat{\mathbf{v}}\mathbf{r}_x - \hat{\mathbf{w}}\mathbf{s}_x)\Gamma(\check{D} - \mathbf{r}_y\check{\mathbf{v}} - \mathbf{s}_y\check{\mathbf{w}}) = 0_{\mathbb{T}}.$$

This fact gives us that the verification equation used by the verifier holds.

$$\begin{aligned}
\hat{C}\Gamma\check{D} &= (\hat{C} - \hat{\mathbf{v}}\mathbf{r}_x - \hat{\mathbf{w}}\mathbf{s}_x)\Gamma\check{D} + \hat{\mathbf{v}}\mathbf{r}_x\Gamma\check{D} + \hat{\mathbf{w}}\mathbf{s}_x\Gamma\check{D} \\
&= (\hat{C} - \hat{\mathbf{v}}\mathbf{r}_x - \hat{\mathbf{w}}\mathbf{s}_x)\Gamma(\check{D} - \mathbf{r}_y\check{\mathbf{v}} - \mathbf{s}_y\check{\mathbf{w}}) + (\hat{C} - \hat{\mathbf{v}}\mathbf{r}_x - \hat{\mathbf{w}}\mathbf{s}_x)\Gamma\mathbf{r}_y\check{\mathbf{v}} + (\hat{C} - \hat{\mathbf{v}}\mathbf{r}_x - \hat{\mathbf{w}}\mathbf{s}_x)\Gamma\mathbf{s}_y\check{\mathbf{w}} \\
&\qquad + \hat{\mathbf{v}}\mathbf{r}_x\Gamma\check{D} + \hat{\mathbf{w}}\mathbf{s}_x\Gamma\check{D} + \hat{\mathbf{v}}(\alpha - \alpha)\check{\mathbf{v}} + \hat{\mathbf{v}}(\beta - \beta)\check{\mathbf{w}} + \hat{\mathbf{w}}(\gamma - \gamma)\check{\mathbf{v}} + \hat{\mathbf{w}}(\delta - \delta)\check{\mathbf{w}} \\
&= 0_{\mathbb{T}} + \Big((\hat{C} - \hat{\mathbf{v}}\mathbf{r}_x - \hat{\mathbf{w}}\mathbf{s}_x)\Gamma\mathbf{r}_y - \hat{\mathbf{v}}\alpha - \hat{\mathbf{w}}\gamma\Big)\check{\mathbf{v}} + \Big((\hat{C} - \hat{\mathbf{v}}\mathbf{r}_x - \hat{\mathbf{w}}\mathbf{s}_x)\Gamma\mathbf{s}_y - \hat{\mathbf{v}}\beta - \hat{\mathbf{w}}\delta\Big)\check{\mathbf{w}} \\
&\qquad + \hat{\mathbf{v}}(\mathbf{r}_x\Gamma\check{D} + \alpha\check{\mathbf{v}} + \beta\check{\mathbf{w}}) + \hat{\mathbf{w}}(\mathbf{s}_x\Gamma\check{D} + \gamma\check{\mathbf{v}} + \delta\check{\mathbf{w}}) \\
&= \hat{\pi}_{\check{v}}\check{\mathbf{v}} + \hat{\pi}_{\check{w}}\check{\mathbf{w}} + \hat{\mathbf{v}}\check{\pi}_{\hat{v}} + \hat{\mathbf{w}}\check{\pi}_{\hat{w}}.
\end{aligned}$$

□

Lemma 4.7.2. *The commit-and-prove scheme has perfect soundness and perfect F-knowledge.*

Proof. The verifier checks that the statement is correctly formatted, the commitments are valid and the proof is well-formed. Finally, the verifier checks that the commitments organized into matrices $\hat{C}$ and $\check{D}$ satisfy

$$\hat{C}\Gamma\check{D} = \hat{\mathbf{v}}\check{\pi}_{\hat{v}} + \hat{\mathbf{w}}\check{\pi}_{\hat{w}} + \hat{\pi}_{\check{v}}\check{\mathbf{v}} + \hat{\pi}_{\check{w}}\check{\mathbf{w}}.$$

Recall $\mathsf{ExtGenCRS}$ returns ξ, ψ such that $\xi\hat{\mathbf{v}} = \xi\hat{\mathbf{w}} = \hat{0}$ and $\check{\mathbf{v}}\psi = \check{\mathbf{w}}\psi = \check{0}$. Multiplying by ξ and ψ on the left and right side respectively, we get

$$\xi\hat{C}\Gamma\check{D}\psi = \xi\hat{\mathbf{v}}\check{\pi}_{\hat{v}}\psi + \xi\hat{\mathbf{w}}\check{\pi}_{\hat{w}}\psi + \xi\hat{\pi}_{\check{v}}\check{\mathbf{v}}\psi + \xi\hat{\pi}_{\check{w}}\check{\mathbf{w}}\psi = 0_{\mathbb{T}}.$$

Observe, $\hat{\mathbf{x}} = \xi \hat{C}$ are the values the extractor $\mathsf{Extract}$ gets from the commitments $(t_{x_1}, \hat{\mathbf{c}}_1), \ldots, (t_{x_m}, \hat{\mathbf{c}}_n)$ and $\check{\mathbf{y}} = \check{D}\psi$ are the values the extractor $\mathsf{Extract}$ gets from the commitments $(t_{y_1}, \check{\mathbf{d}}_1), \ldots, (t_{y_n}, \check{\mathbf{d}}_n)$. The extracted values from the commitments therefore satisfy

$$\hat{\mathbf{x}}\Gamma\check{\mathbf{y}} = 0_{\mathbb{T}}.$$

If the equation is a pairing product equation, this directly gives us both perfect soundness and perfect F-knowledge.

If the equation is a multi-scalar multiplication equation in $\check{\mathbb{H}}$, then the committed value is $\mathbf{x} = \log_{\hat{g}} \hat{\mathbf{x}}$. Then the opened values satisfy $\mathbf{x}\Gamma\check{\mathbf{y}} = \check{0}$, which gives us perfect soundness and perfect F-knowledge. The case of a multi-scalar multiplication in $\hat{\mathbb{G}}$ is similar. Finally, for quadratic equations over $\mathbb{Z}_p$ the committed values are $\mathbf{x} = \log_{\hat{g}} \hat{\mathbf{x}}$, $\mathbf{y} = \log_{\check{h}} \check{\mathbf{y}}$ and they satisfy $\mathbf{x}\Gamma\mathbf{y} = 0$, which gives us perfect soundness and perfect F-knowledge.

□

Lemma 4.7.3. *The commit-and-prove scheme is composable zero-knowledge if the SXDH assumption holds relative to $\mathcal{PG}$.*

Proof. We first recall what the GenCRS and $\mathsf{SimGenCRS}$ algorithms are. Both algorithms output a commitment key which specifies elements $\hat{\mathbf{v}}, \hat{\mathbf{w}}, \hat{\mathbf{u}}, \check{\mathbf{v}}, \check{\mathbf{w}}, \check{\mathbf{u}}$. The only difference between GenCRS and $\mathsf{SimGenCRS}$ (besides the fact that $\mathsf{SimGenCRS}$ outputs a simulation trapdoor) is that in GenCRS, the elements are of the form $\hat{\mathbf{v}} = (\hat{g}, \xi\hat{g})^{\top}$, $\check{\mathbf{v}} = (\check{h}, \check{h}\psi)$, $\hat{\mathbf{w}} = \rho\hat{\mathbf{v}}$, $\check{\mathbf{w}} = \sigma\check{\mathbf{v}}$, $\hat{\mathbf{u}} = \rho\hat{\mathbf{v}} + \mathbf{e}^{\top}\hat{g}$, $\check{\mathbf{u}} = \sigma\check{\mathbf{v}} + \mathbf{e}\check{h}$, whereas in $\mathsf{SimGenCRS}$ the elements are of the form $\hat{\mathbf{v}} = (\hat{g}, \xi\hat{g})^{\top}$, $\check{\mathbf{v}} = (\check{h}, \psi\check{h})$, $\hat{\mathbf{w}} = \rho\hat{\mathbf{v}} - \mathbf{e}^{\top}\hat{g}$, $\check{\mathbf{w}} = \sigma\check{\mathbf{v}} - \mathbf{e}\check{h}$, $\hat{\mathbf{u}} = \rho\hat{\mathbf{v}}$, $\check{\mathbf{u}} = \sigma\check{\mathbf{v}}$.

Under the SXDH assumption, the distribution of the elements $\hat{\mathbf{v}}, \hat{\mathbf{w}}, \hat{\mathbf{u}}, \check{\mathbf{v}}, \check{\mathbf{w}}, \check{\mathbf{u}}$ generated by GenCRS is computationally indistinguishable from the distribution of the elements $\hat{\mathbf{v}}, \hat{\mathbf{w}}, \hat{\mathbf{u}}, \check{\mathbf{v}}, \check{\mathbf{w}}, \check{\mathbf{u}}$ generated by $\mathsf{SimGenCRS}$. This shows that the first requirement of composable zero-knowledge is satisfied.

For the second requirement, we first give a high-level description of the simulator in Fig. 4.10.

- For elements in the witness such that $t \in \{\mathrm{com}_{\hat{\mathbb{G}}}, \mathrm{sca}_{\hat{\mathbb{G}}}, \mathrm{com}_{\check{\mathbb{H}}}, \mathrm{sca}_{\check{\mathbb{H}}}\}$, the simulator will create commitments to $m = \hat{0}, m = \check{0}$ or $m = 0$ according to the type. These commitments are perfectly hiding under the commitment key output by $\mathsf{SimGenCRS}$.

- For elements in the witness such that $t \in \{\mathrm{base}_{\hat{\mathbb{G}}}, \mathrm{unit}_{\hat{\mathbb{G}}}, \mathrm{base}_{\check{\mathbb{H}}}, \mathrm{unit}_{\check{\mathbb{H}}}\}$, the simulator will use the simulation trapdoor tk to be able to treat them as commitments to $\hat{0}, \check{0}$ or 0 according to t.

- For elements in the witness such that $t \in \{\mathrm{enc}_{\hat{\mathbb{G}}}, \mathrm{enc}_{\check{\mathbb{H}}}\}$, the commitments are computed as encryptions of $\hat{0}$ and $\check{0}$ respectively.

- Having obtained satisfying openings of the commitments the simulator creates the proof as an honest prover would do with those values.

We now argue that a simulated proof is computationally indistinguishable from an honestly generated proof when the commitment key is output using $\mathsf{SimGenCRS}$. To show that, we define three games and use a hybrid argument.

- Game 1 is the game where the commitment key is output using $\mathsf{SimGenCRS}$ and the proof is done using a valid witness.
- Game 2 is as Game 1, but instead of using a valid witness, commitments with $t \in \{\mathrm{com}_{\hat{\mathbb{G}}}, \mathrm{sca}_{\hat{\mathbb{G}}}, \mathrm{com}_{\check{\mathbb{H}}}, \mathrm{sca}_{\check{\mathbb{H}}}\}$ are done to $m = \hat{0}, m = \check{0}$ or $m = 0$, and commitments with $t \in \{\mathrm{base}_{\hat{\mathbb{G}}}, \mathrm{unit}_{\hat{\mathbb{G}}}, \mathrm{base}_{\check{\mathbb{H}}}, \mathrm{unit}_{\check{\mathbb{H}}}\}$ are equivocated as commitments to $\hat{0}, \check{0}$ or 0, according to t and T as described in Fig. 4.10.
- Game 3 is as Game 2, but commitments with $t \in \{\mathrm{enc}_{\hat{\mathbb{G}}}, \mathrm{enc}_{\check{\mathbb{H}}}\}$ are changed for encryptions of $\hat{0}$, $\check{0}$ according to t. Note that Game 3 is equivalent to the simulation.

Game 1 and Game 2 are perfectly indistinguishable due to the fact that commitments with $t \in \{\mathrm{com}_{\hat{\mathbb{G}}}, \mathrm{sca}_{\hat{\mathbb{G}}}, \mathrm{com}_{\check{\mathbb{H}}}, \mathrm{sca}_{\check{\mathbb{H}}}\}$ are perfectly hiding and the uniformity of the randomized proofs, which we prove in Lemma 4.7.4. Equivocating the commitments with $t \in \{\mathrm{base}_{\hat{\mathbb{G}}}, \mathrm{unit}_{\hat{\mathbb{G}}}, \mathrm{base}_{\check{\mathbb{H}}}, \mathrm{unit}_{\check{\mathbb{H}}}\}$ as commitments to $\hat{0}, \check{0}$ or 0 only changes the proofs and not the commitments themselves, so this case is also indistinguishable by Lemma 4.7.4.

Game 2 and Game 3 are computationally indistinguishable due to the IND-CPA security of ElGamal. Note that, in Game 2 and Game 3, the equation is satisfied for any value of the elements in the witness such that $t \in \{\mathrm{enc}_{\hat{\mathbb{G}}}, \mathrm{enc}_{\check{\mathbb{H}}}\}$. Actually, for each element $\hat{x}_i$ and $\check{y}_j$ we can just get the corresponding ciphertext $\hat{\mathbf{c}}_i$ or $\check{\mathbf{d}}_j$ and build the proof on top of it. We can do this because all the terms of the proof which have to be computed using the ciphertext's randomness will be 0 as the element paired to the randomness is going to be 0 due to the restrictions on the equation[2]. This allows us to reduce the computational indistinguishability of Game 2 and Game 3 to the IND-CPA security of ElGamal. □

Lemma 4.7.4. *Proofs are uniformly random over solutions to the verification equations when the commitment key is generated using* $\mathsf{SimGenCRS}$.

Proof. In all games from Game 1 to Game 3, all witnesses will yield proofs satisfying the verification equation by perfect completeness. Let us now argue that these proofs are uniformly random over the possible solutions to the verification equation. Observe that, as the commitment key is generated using $\mathsf{SimGenCRS}$, we have that $\hat{\mathbf{v}}, \hat{\mathbf{w}}$ generate $\hat{\mathbb{G}}^{1\times 2}$, $\check{\mathbf{v}}, \check{\mathbf{w}}$ generate $\check{\mathbb{H}}^{2\times 1}$, $\hat{\mathbf{u}} \in \langle \hat{\mathbf{v}} \rangle$ and $\check{\mathbf{u}} \in \langle \check{\mathbf{v}} \rangle$. From now on, let us define the pre-proofs as those proofs that satisfy the verification equation but haven't been randomized yet, for instance, $\hat{\pi}'_{\check{v}} = \mathbf{e}^{\top} \hat{\mathbf{x}} \Gamma \mathbf{r}_y$.

[2] Strictly speaking, if the proofs are computed using the algorithm given in Fig. 4.10, then this argument is only true for encryptions of elements in $\check{\mathbb{H}}$. However, the proofs can be written in an alternative way by changing the role of $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$ so that the same argument holds for encryptions of elements in $\hat{\mathbb{G}}$

We will prove either that there is only one proof which is accepted by the verification equation or that when there are many possible proofs both honestly generated and simulated proofs will be uniformly distributed among all valid proofs, i.e., among all solutions to the verification equations. In the first case, by using the techniques used in [GS12] it is easy to see that when $\check{\pi}_{\hat{v}}, \check{\pi}_{\hat{w}}$ (resp. $\hat{\pi}_{\check{v}}, \hat{\pi}_{\check{w}}$) are 0 there is only one value for $\hat{\pi}_{\check{v}}, \hat{\pi}_{\check{w}}$ (resp. $\check{\pi}_{\hat{v}}, \check{\pi}_{\hat{w}}$) which satisfies the verification equation. In the latter case, we will show that $\check{\pi}_{\hat{v}}, \check{\pi}_{\hat{w}}$ (resp. $\hat{\pi}_{\check{v}}, \hat{\pi}_{\check{w}}$) are uniformly random, and using the techniques from [GS12] it is easy to see that for fixed values for $\check{\pi}_{\hat{v}}, \check{\pi}_{\hat{w}}$ (resp. $\hat{\pi}_{\check{v}}, \hat{\pi}_{\check{w}}$) there is only one value for $\hat{\pi}_{\check{v}}, \hat{\pi}_{\check{w}}$ (resp. $\check{\pi}_{\hat{v}}, \check{\pi}_{\hat{w}}$) which satisfies the verification equation. If some of $\check{\pi}'_{\hat{v}}, \check{\pi}'_{\hat{w}}, \hat{\pi}'_{\check{v}}, \hat{\pi}'_{\check{w}}$ are already 0 due to the type of equation, we will not randomize them. The reason is that due to the restrictions on the equation types, if for an honestly generated proof some of $\check{\pi}'_{\hat{v}}, \check{\pi}'_{\hat{w}}, \hat{\pi}'_{\check{v}}, \hat{\pi}'_{\check{w}}$ are 0 then the zero-knowledge simulator will be able to construct simulated proofs such that the same parts of the proof are 0. Therefore, all valid proofs will be such that the same parts of the proof are 0, so these components will not need to be randomized.

For each type of equation, we now detail whether $\check{\pi}_{\hat{v}}, \check{\pi}_{\hat{w}}$ (or $\hat{\pi}_{\check{v}}, \hat{\pi}_{\check{w}}$) are 0 or whether $\check{\pi}_{\hat{v}}, \check{\pi}_{\hat{w}}$ (or $\hat{\pi}_{\check{v}}, \hat{\pi}_{\check{w}}$) are uniformly random, where we assume that $\alpha, \beta, \gamma, \delta \leftarrow \mathbb{Z}_p$:

For $T =$ PPE, $\check{\pi}_{\hat{v}} = \check{\pi}'_{\hat{v}} + \alpha\check{\mathbf{v}} + \beta\check{\mathbf{w}}$, $\check{\pi}_{\hat{w}} = \check{\pi}'_{\hat{w}} + \gamma\check{\mathbf{v}} + \delta\check{\mathbf{w}}$ are uniformly distributed. For $T =$ PEnc$_{\hat{\mathbb{G}}}$, $\check{\pi}_{\hat{v}} = \check{\pi}'_{\hat{v}} + \alpha\check{\mathbf{v}} + \beta\check{\mathbf{w}}$ is uniformly distributed and $\check{\pi}_{\hat{w}} = 0$ (similarly for $T =$ PEnc$_{\check{\mathbb{H}}}$). For $T =$ PConst$_{\hat{\mathbb{G}}}$, $\check{\pi}_{\hat{v}} = \check{\pi}_{\hat{w}} = 0$ (similarly for $T =$ PConst$_{\check{\mathbb{H}}}$).

For $T =$ ME$_{\hat{\mathbb{G}}}$, $\hat{\pi}_{\check{v}} = \hat{\pi}'_{\check{v}} + \alpha\hat{\mathbf{v}} + \beta\hat{\mathbf{w}}$ is uniformly distributed and $\hat{\pi}_{\check{w}} = 0$. For $T =$ MEnc$_{\hat{\mathbb{G}}}$, $\check{\pi}_{\hat{v}} = \check{\pi}'_{\hat{v}} + \alpha\check{\mathbf{v}}$ is uniformly distributed and $\check{\pi}_{\hat{w}} = 0$. For $T =$ MConst$_{\hat{\mathbb{G}}}$, $\check{\pi}_{\hat{v}}, \check{\pi}_{\hat{w}} = 0$. For $T =$ MLin$_{\hat{\mathbb{G}}}$, $\hat{\pi}_{\check{v}}, \hat{\pi}_{\check{w}} = 0$. (Similarly, for $T \in \{\mathrm{ME}_{\check{\mathbb{H}}}, \mathrm{MEnc}_{\check{\mathbb{H}}}, \mathrm{MConst}_{\check{\mathbb{H}}}, \mathrm{MLin}_{\check{\mathbb{H}}}\}$).

For $T =$ QE, $\hat{\pi}_{\check{v}} = \hat{\pi}'_{\check{v}} + \alpha\hat{\mathbf{v}}$ is uniformly distributed and $\hat{\pi}_{\check{w}} = 0$. For $T =$ QConst$_{\hat{\mathbb{G}}}$, $\check{\pi}_{\hat{v}} = \check{\pi}_{\hat{w}} = 0$ (similarly for $T =$ QConst$_{\check{\mathbb{H}}}$). □

4.8 NIZK proofs with prover-chosen CRS

In Groth-Sahai proofs, the prover uses a common reference string shared between the prover and the verifier to construct NIZK proofs. We can improve efficiency by letting the prover choose her own common reference string, which we will refer to as her public key. The public key has to be created as a perfectly binding key, otherwise the soundness of the NIZK proof could be easily broken by the prover. The prover will then have to prove to the verifier that her public key has been created as a perfectly binding key. To do that the prover will give an NIZK proof using the shared common reference string. The definitions for commit-and-prove schemes with prover-chosen CRS are given in Sec. 4.8.1. In Secs. 4.8.2 to 4.8.4 we explain how the prover creates her public key, proves its well-formedness and we detail what the efficiency improvement obtained is. Finally, in Sec. 4.8.5 we prove the security of our scheme.

4.8.1 Definitions for prover-chosen CRS commit-and-prove schemes

The definitions given in Sec. 4.2 are not suitable for the scenario where the prover chooses her own common reference string. The reason is that the prover will only communicate her public key pk to the verifier once, and the public key pk will be used to generate and verify commitments and proofs.

Therefore, we define extended commit-and-prove schemes for a relation R, which capture the fact that the prover chooses her own public key. We will only consider the common reference string model, where the prover makes a well-formedness NIZK proof for her pk by using the shared common reference string between the prover and the verifier. In particular, we are not considering the multi-string model [GO14] nor the scenario where the prover gets a certificate on her public key. The definitions can be naturally extended to those settings.

The definitions share many elements with the definitions given in Sec. 4.2. We may commit to different values $w_1, \ldots, w_N$ and prove for different statements x that a subset of the committed values $w = (w_{i_1}, \ldots, w_{i_n})$ constitute a witness for $x \in L_{gk}$, i.e., $(gk, x, w) \in R$. We will also divide each committed value into two parts $w_i = (t_i, m_i)$.

An extended commit-and-prove scheme ECP $(\mathsf{Setup}, \mathsf{GenCRS}, \mathsf{CreatePK}, \mathsf{VerifyPK}, \mathsf{Commit}, \mathsf{Prove}, \mathsf{VerifyProof})$ consists of six algorithms. The algorithms $\mathsf{Setup}, \mathsf{GenCRS}, \mathsf{CreatePK}, \mathsf{Prove}$ are probabilistic and the algorithms $\mathsf{VerifyPK}, \mathsf{Commit}, \mathsf{VerifyProof}$ are deterministic.

$\mathsf{Setup}(1^\lambda)$**:** On input a security parameter 1^λ outputs a setup gk. The setup specifies a message space $\mathfrak{M}_{gk}$, a randomness space $\mathfrak{R}_{gk}$ and a commitment space $\mathfrak{C}_{gk}$. Membership of either space can be decided efficiently.

$\mathsf{GenCRS}(gk)$**:** Generates a commitment key ck.

$\mathsf{CreatePK}(gk, ck)$**:** Given a setup gk and a commitment key ck, returns a public key pk, a secret key sk and a well-formedness proof π_{pk}.

$\mathsf{VerifyPK}(gk, ck, pk, \pi_{pk})$**:** Given a setup gk, a commitment key ck, a public key pk and a well-formedness proof π_{pk} returns 1 (accept) or 0 (reject).

$\mathsf{Commit}(gk, ck, sk, t, m; r)$**:** Given a setup gk, a commitment key ck, a secret key sk, a message $(t, m) \in \mathfrak{M}_{gk}$ and randomness r such that $(t, r) \in \mathfrak{R}_{gk}$ returns a commitment c such that $(t, c) \in \mathfrak{C}_{gk}$.

$\mathsf{Prove}(gk, ck, sk, x, (t_1, m_1, r_1), \ldots, (t_n, m_n, r_n))$**:** Given a setup gk, a commitment key ck, a secret key sk, a statement x, and commitment openings such that $(t_i, m_i) \in \mathfrak{M}_{gk}, (t_i, r_i) \in \mathfrak{R}_{gk}$ and $(gk, ck, x, t_1, m_1, \ldots, t_n, m_n) \in R$ returns a proof π.

$\mathsf{VerifyProof}(gk, ck, pk, x, (t_1, c_1), \ldots, (t_n, c_n), \pi)$**:** Given a setup gk, a commitment key ck, a public key pk, a statement x, a proof π and commitments $(t_i, c_i) \in \mathfrak{C}_{gk}$ returns 1 (accept) or 0 (reject).

Definition 4.8.1 (Perfect correctness)**.** *The extended commit-and-prove system* ECP *is (perfectly) correct if for all adversaries $\mathcal{A}$*

$$\Pr\left[\begin{array}{l} gk \leftarrow \mathsf{Setup}(1^\lambda); ck \leftarrow \mathsf{GenCRS}(gk)\ ; \\ (pk, sk, \pi_{pk}) \leftarrow \mathsf{CreatePK}(gk, ck); (x, w_1, r_1, \ldots, w_n, r_n) \leftarrow \mathcal{A}(gk, ck, pk, sk, \pi_{pk})\ ; \\ c_i \leftarrow \mathsf{Commit}(gk, ck, sk, w_i; r_i)\ ; \pi \leftarrow \mathsf{Prove}(gk, ck, sk, x, w_1, r_1, \ldots, w_n, r_n) : \\ \mathsf{VerifyPK}(gk, ck, pk, \pi_{pk}) \cdot \mathsf{VerifyProof}(gk, ck, pk, x, (t_1, c_1), \ldots, (t_n, c_n), \pi) = 1 \end{array}\right] = 1,$$

where $\mathcal{A}$ outputs w_i, r_i such that $w_i = (t_i, m_i) \in \mathfrak{M}_{gk}, (t_i, r_i) \in \mathfrak{R}_{gk}$ and $(gk, ck, x, w_1, \ldots, w_n) \in R$.

Definition 4.8.2 (Perfect soundness)**.** *The extended commit-and-prove system* ECP *is (perfectly) sound if there exists a deterministic (unbounded) opening algorithm* Open *such that for all adversaries $\mathcal{A}$*

$$\Pr\left[\begin{array}{l} gk \leftarrow \mathsf{Setup}(1^\lambda); ck \leftarrow \mathsf{GenCRS}(gk)\ ; \\ (pk, \pi_{pk}, x, t_1, c_1, \ldots, t_n, c_n, \pi) \leftarrow \mathcal{A}(gk, ck)\ ;\ m_i \leftarrow \mathrm{Open}(gk, ck, t_i, c_i) : \\ \mathsf{VerifyPK}(gk, ck, pk, \pi_{pk}) \cdot \mathsf{VerifyProof}(gk, ck, pk, x, t_1, c_1, \ldots, t_n, c_n, \pi) = 0 \\ \vee\ (gk, ck, x, (t_1, m_1), \ldots, (t_n, m_n)) \in R \end{array}\right] = 1.$$

Perfect F-knowledge is defined exactly as for commit-and-prove schemes. This means that we will be using a global extraction key associated with ck for all commitments regardless of which public keys are used to generate the commitments.

Definition 4.8.3 (Composable zero-knowledge)**.** *The extended commit-and-prove system* ECP *is (computationally) composable zero-knowledge if there exist some p.p.t. algorithms* SimGenCRS, SimCreatePK, SimCommit, SimProve *such that for all non-uniform polynomial time stateful interactive adversaries $\mathcal{A}$*

$$\Pr\left[\begin{array}{l} gk \leftarrow \mathsf{Setup}(1^\lambda); ck \leftarrow \mathsf{GenCRS}(gk) : \\ \mathcal{A}(gk, ck) = 1 \end{array}\right] \approx \Pr\left[\begin{array}{l} gk \leftarrow \mathsf{Setup}(1^\lambda); ck \leftarrow \mathsf{SimGenCRS}(gk) : \\ \mathcal{A}(gk, ck) = 1 \end{array}\right]$$

and

$$\Pr\left[\begin{array}{l} gk \leftarrow \mathsf{Setup}(1^\lambda); (ck, tk) \leftarrow \mathsf{SimGenCRS}(gk); (pk, sk, \pi_{pk}) \leftarrow \mathsf{CreatePK}(gk, ck) : \\ \mathcal{A}(gk, ck, tk, pk, \pi_{pk}) = 1 \end{array}\right]$$

$$\approx \Pr\left[\begin{array}{l} gk \leftarrow \mathsf{Setup}(1^\lambda); (ck, tk) \leftarrow \mathsf{SimGenCRS}(gk); (pk, sk, \pi_{pk}) \leftarrow \mathsf{SimCreatePK}(gk, tk) : \\ \mathcal{A}(gk, ck, tk, pk, \pi_{pk}) = 1 \end{array}\right],$$

and

$$\Pr\left[\begin{array}{l} gk \leftarrow \mathsf{Setup}(1^\lambda); (ck, tk) \leftarrow \mathsf{SimGenCRS}(gk); (pk, sk, \pi_{pk}) \leftarrow \mathsf{SimCreatePK}(gk, tk); \\ (x, i_1, \ldots, i_n) \leftarrow \mathcal{A}^{\mathsf{Commit}(gk,ck,sk,\cdot)}(gk, ck, tk, pk, sk, \pi_{pk}); \\ \pi \leftarrow \mathsf{Prove}(gk, ck, sk, x, w_{i_1}, r_{i_1}, \ldots, w_{i_n}, r_{i_n}) : \\ \mathcal{A}(\pi) = 1 \end{array}\right]$$

$$\approx \Pr\left[\begin{array}{l} gk \leftarrow \mathsf{Setup}(1^\lambda); (ck, tk) \leftarrow \mathsf{SimGenCRS}(gk); (pk, sk, \pi_{pk}) \leftarrow \mathsf{SimCreatePK}(gk, tk); \\ (x, i_1, \ldots, i_n) \leftarrow \mathcal{A}^{\mathsf{SimCommit}(gk,ck,sk,\cdot)}(gk, ck, tk, pk, sk, \pi_{pk}); \\ \pi \leftarrow \mathsf{SimProve}(gk, ck, sk, x, t_{i_1}, s_{i_1}, \ldots, t_{i_n}, s_{i_n}) : \\ \mathcal{A}(\pi) = 1 \end{array}\right],$$

where

- $\mathsf{Commit}(gk, ck, sk, \cdot)$ *on* $w_i = (t_i, m_i) \in \mathfrak{M}_{gk}$ *picks uniformly random* r_i *such that* $(t_i, r_i) \in \mathfrak{R}_{gk}$ *and returns* $c_i = \mathsf{Commit}(gk, ck, sk, w_i; r_i)$
- $\mathsf{SimCommit}(gk, ck, sk, \cdot)$ *on* $w_i = (t_i, m_i) \in \mathfrak{M}_{gk}$ *runs* $(c_i, s_i) \leftarrow \mathsf{SimCommit}(gk, ck, sk, t_i)$ *and returns* c_i*, where* s_i *is some auxiliary information used to construct simulated proofs.*
- $\mathcal{A}$ *picks* $(x, i_1, \ldots, i_n)$ *such that* $(gk, ck, x, w_{i_1}, \ldots, w_{i_n}) \in R$

4.8.2 Creating the public key

As in a regular commit-and-prove system, the setup algorithm Setup takes as input a security parameter 1^λ and runs $\mathcal{PG}(1^\lambda)$ to return a bilinear group $(p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h})$. Like commitment keys, public keys can be created in two ways: they can either be perfectly binding or perfectly hiding. These two types of keys are computationally indistinguishable if the SXDH assumption holds. As we already argued, we will require the prover to create her public key in a perfectly binding way. However, the zero-knowledge simulator will create a perfectly hiding public key and simulate the NIZK proof for well-formedness.

As shown in Fig. 4.11, the public key is created in a similar way to how the commitment key is created. The main difference is that we allow the prover to reuse the elements $\hat{\mathbf{v}}, \check{\mathbf{v}}$. This both reduces the size of the public key and also ensures that the prover's commitments are extractable even when using her own key.

Once the prover has created her pair of public key and secret key, she has to compute an NIZK proof to show that her pk is perfectly binding. A valid public key is defined by the existence of some ρ_P, σ_P such that $\hat{\mathbf{w}}_P = \rho_P \hat{\mathbf{v}}$ and $\check{\mathbf{w}}_P = \sigma_P \check{\mathbf{v}}$, which can be written as two equations of type $\mathrm{MConst}_{\hat{\mathbb{G}}}$ involving public elements in $\hat{\mathbb{G}}$ and a secret ρ_P committed in $\check{\mathbb{H}}$, and two equations of type $\mathrm{MConst}_{\check{\mathbb{H}}}$ involving public elements in $\check{\mathbb{H}}$ and a secret σ_P committed in $\hat{\mathbb{G}}$. These are simple statements that each have a proof consisting of a single group element. In Fig. 4.12 we give the exact NIZK proofs that have to be computed. The total cost of communicating the public key, which is determined by $\hat{\mathbf{w}}_P, \check{\mathbf{w}}_P$, the two commitments to ρ_P and

```
GenPK$(gk, ck)$
    $\rho_P \leftarrow \mathbb{Z}_p$    $\sigma_P \leftarrow \mathbb{Z}_p$
    $\hat{\mathbf{v}}_P \leftarrow \hat{\mathbf{v}}$    $\check{\mathbf{v}}_P \leftarrow \check{\mathbf{v}}$
    $\hat{\mathbf{w}}_P \leftarrow \rho_P \hat{\mathbf{v}}_P$    $\check{\mathbf{w}}_P \leftarrow \sigma_P \check{\mathbf{v}}_P$
    $\hat{\mathbf{u}}_P \leftarrow \hat{\mathbf{w}}_P + (\hat{0}, \hat{g})^\top$    $\check{\mathbf{u}}_P \leftarrow \check{\mathbf{w}}_P + (\check{0}, \check{h})$
  $pk \leftarrow (\hat{\mathbf{u}}_P, \hat{\mathbf{v}}_P, \hat{\mathbf{w}}_P, \check{\mathbf{u}}_P, \check{\mathbf{v}}_P, \check{\mathbf{w}}_P)$
  $sk \leftarrow (pk, \rho_P, \sigma_P)$
  Return $(pk, sk)$
```

```
SimGenPK$(gk, ck)$
    $\rho_P \leftarrow \mathbb{Z}_p$    $\sigma_P \leftarrow \mathbb{Z}_p$
    $\hat{\mathbf{v}}_P \leftarrow \hat{\mathbf{v}}$    $\check{\mathbf{v}}_P \leftarrow \check{\mathbf{v}}$
    $\hat{\mathbf{w}}_P \leftarrow \rho_P \hat{\mathbf{v}}_P - (\hat{0}, \hat{g})^\top$    $\check{\mathbf{w}}_P \leftarrow \sigma_P \check{\mathbf{v}}_P - (\check{0}, \check{h})$
    $\hat{\mathbf{u}}_P \leftarrow \hat{\mathbf{w}}_P + (\hat{0}, \hat{g})^\top$    $\check{\mathbf{u}}_P \leftarrow \check{\mathbf{w}}_P + (\check{0}, \check{h})$
  $pk \leftarrow (\hat{\mathbf{u}}_P, \hat{\mathbf{v}}_P, \hat{\mathbf{w}}_P, \check{\mathbf{u}}_P, \check{\mathbf{v}}_P, \check{\mathbf{w}}_P)$
  $sk \leftarrow (pk, \rho_P, \sigma_P)$
  Return $(pk, sk)$
```

Figure 4.11: Public key generator algorithms

σ_P and the four NIZK proofs is 12 group elements. Since we are using a commit-and-prove scheme we can consider this as a one-off cost for each verifier engaging with the prover after which the public key may be used for many commitments and proofs.

4.8.3 Proving the well-formedness of a prover-chosen CRS

As explained in Sec. 4.8.2, once the prover has chosen its own CRS a proof of well-formedness has to be given in order to guarantee soundness. This is, the prover has to show that the values $\hat{\mathbf{v}}_P, \hat{\mathbf{w}}_P, \check{\mathbf{v}}_P, \check{\mathbf{w}}_P$ satisfying

$$\hat{\mathbf{w}}_P - \hat{\mathbf{v}}_P \rho_P = \hat{\mathbf{0}} \qquad \text{and} \qquad \check{\mathbf{w}}_P - \sigma_P \check{\mathbf{v}}_P = \check{\mathbf{0}},$$

for some secret values ρ_P, σ_P. To prove that these relations are satisfied, the prover can use Groth-Sahai proofs. In Fig. 4.12 we detail what proofs need to be done by the prover to prove the well-formedness of her pk, writing J^{ij} for the 2×2 matrix such that all its elements are 0 except the element in row i and column j, which takes the value 1. Note that these are ordinary Groth-Sahai proofs, which can be verified using the commitment key ck.

4.8.4 Computing commitments and NIZK proofs

Once the prover has created her public key pk and has proven its well-formedness, she can make commitments and prove statements using pk instead of ck. The commitments and proofs are created and verified in exactly the same way as described in Fig. 4.3 and Fig. 4.8, but the number of scalar multiplications needed to compute commitments and NIZK proofs can be reduced using her knowledge of the discrete logarithms in sk. We have for instance

$$\hat{\mathbf{c}} = \mathbf{e}^\top \hat{x} + \hat{\mathbf{v}} r + \hat{\mathbf{w}}_P s = \mathbf{e}^\top \hat{x} + \hat{\mathbf{v}}(r + \rho_P s),$$

```
$\mathsf{ProvePK}(gk, ck, pk, sk)$
Parse $\hat{\mathbf{v}}_P, \hat{\mathbf{w}}_P, \check{\mathbf{v}}_P, \check{\mathbf{w}}_P$ as $((\hat{\mathbf{v}}_P)_1, (\hat{\mathbf{v}}_P)_2 = \hat{g}), ((\hat{\mathbf{w}}_P)_1, (\hat{\mathbf{w}}_P)_2), ((\check{\mathbf{v}}_P)_1, (\check{\mathbf{v}}_P)_2 = \check{h}), ((\check{\mathbf{w}}_P)_1, (\check{\mathbf{w}}_P)_2))$
Compute the commitments $\check{d}_\rho \leftarrow \rho_P \check{\mathbf{u}}_P + r_\rho \check{\mathbf{v}}; \hat{c}_\sigma \leftarrow \hat{\mathbf{u}}_P \sigma_P + \hat{\mathbf{v}} r_\sigma;$ where $r_\rho, r_\sigma \leftarrow \mathbb{Z}_p$
Define the following variables and types:
$r_{x_1}, s_{x_1}, r_{x_2}, s_{x_2}, r_{y_1}, s_{y_1}, s_{y_2} \leftarrow 0;\ r_{y_2} \leftarrow r_\rho$
  $t_{x_1} = (\mathrm{pub}_{\hat{\mathbb{G}}}, (\hat{\mathbf{w}}_P)_1),\quad x_1 = (\hat{\mathbf{w}}_P)_1,\quad t_{x_2} = (\mathrm{pub}_{\hat{\mathbb{G}}}, (\hat{\mathbf{v}}_P)_1),\quad x_2 = (\hat{\mathbf{v}}_P)_1$
  $t_{y_1} = \mathrm{unit}_{\check{\mathbb{H}}},\quad y_1 = 1,\quad t_{y_2} = \mathrm{sca}_{\check{\mathbb{H}}},\quad y_2 = \rho_P$
$\pi_1 \leftarrow \mathsf{Prove}(gk, ck, \mathrm{MConst}_{\hat{\mathbb{G}}}, (J^{11} - J^{22}), \{(t_{x_i}, x_i, (r_{x_i}, s_{x_i}))\}_{i=1}^{2}, \{(t_{y_j}, y_j, (r_{y_j}, s_{y_j}))\}_{j=1}^{2})$
Redefine the following variables and types:
$r_{x_1}, s_{x_1}, r_{x_2}, s_{x_2}, r_{y_1}, s_{y_1}, s_{y_2} \leftarrow 0;\ r_{y_2} \leftarrow r_\rho$
  $t_{x_1} = (\mathrm{pub}_{\hat{\mathbb{G}}}, (\hat{\mathbf{w}}_P)_2),\quad x_1 = (\hat{\mathbf{w}}_P)_2,\quad t_{x_2} = \mathrm{base}_{\hat{\mathbb{G}}},\quad x_2 = \hat{g}$
  $t_{y_1} = \mathrm{unit}_{\check{\mathbb{H}}},\quad y_1 = 1,\quad t_{y_2} = \mathrm{sca}_{\check{\mathbb{H}}},\quad y_2 = \rho_P$
$\pi_2 \leftarrow \mathsf{Prove}(gk, ck, \mathrm{MConst}_{\hat{\mathbb{G}}}, (J^{11} - J^{22}), \{(t_{x_i}, x_i, (r_{x_i}, s_{x_i}))\}_{i=1}^{2}, \{(t_{y_j}, y_j, (r_{y_j}, s_{y_j}))\}_{j=1}^{2})$
Redefine the following variables and types:
$r_{x_1}, s_{x_1}, s_{x_2}, r_{y_1}, s_{y_1}, r_{y_2}, s_{y_2} \leftarrow 0;\ r_{x_2} \leftarrow r_\sigma$
  $t_{x_1} = \mathrm{unit}_{\hat{\mathbb{G}}},\quad x_1 = 1,\quad t_{x_2} = \mathrm{sca}_{\hat{\mathbb{G}}},\quad x_2 = \sigma_P$
  $t_{y_1} = (\mathrm{pub}_{\check{\mathbb{H}}}, (\check{\mathbf{w}}_P)_1),\quad y_1 = (\check{\mathbf{w}}_P)_1,\quad t_{y_2} = (\mathrm{pub}_{\check{\mathbb{H}}}, (\check{\mathbf{v}}_P)_1),\quad y_2 = (\check{\mathbf{v}}_P)_1$
$\pi_3 \leftarrow \mathsf{Prove}(gk, ck, \mathrm{MConst}_{\check{\mathbb{H}}}, (J^{11} - J^{22}), \{(t_{x_i}, x_i, (r_{x_i}, s_{x_i}))\}_{i=1}^{2}, \{(t_{y_j}, y_j, (r_{y_j}, s_{y_j}))\}_{j=1}^{2})$
Redefine the following variables and types:
$r_{x_1}, s_{x_1}, s_{x_2}, r_{y_1}, s_{y_1}, r_{y_2}, s_{y_2} \leftarrow 0;\ r_{x_2} \leftarrow r_\sigma$
  $t_{x_1} = \mathrm{unit}_{\hat{\mathbb{G}}},\quad x_1 = 1,\quad t_{x_2} = \mathrm{sca}_{\hat{\mathbb{G}}},\quad x_2 = \sigma_P$
  $t_{y_1} = (\mathrm{pub}_{\check{\mathbb{H}}}, (\check{\mathbf{w}}_P)_2),\quad y_1 = (\check{\mathbf{w}}_P)_2,\quad t_{y_2} = \mathrm{base}_{\check{\mathbb{H}}},\quad y_2 = \check{h}$
$\pi_4 \leftarrow \mathsf{Prove}(gk, ck, \mathrm{MConst}_{\check{\mathbb{H}}}, (J^{11} - J^{22}), \{(t_{x_i}, x_i, (r_{x_i}, s_{x_i}))\}_{i=1}^{2}, \{(t_{y_j}, y_j, (r_{y_j}, s_{y_j}))\}_{j=1}^{2})$
Return $\pi_{pk} = (\hat{c}_\rho, \check{d}_\sigma, \pi_1, \pi_2, \pi_3, \pi_4)$
```

Figure 4.12: Algorithm to prove well-formedness of the public key

	GS commitments		This work	
t	$\hat{\mathbb{G}}$	$\check{\mathbb{H}}$	$\hat{\mathbb{G}}$	$\check{\mathbb{H}}$
$\mathrm{com}_{\hat{\mathbb{G}}}$	4	0	2	0
$\mathrm{com}_{\check{\mathbb{H}}}$	0	4	0	2
$\mathrm{enc}_{\hat{\mathbb{G}}}$	Undefined		2	0
$\mathrm{enc}_{\check{\mathbb{H}}}$	Undefined		0	2
$\mathrm{sca}_{\hat{\mathbb{G}}}$	4	0	2	0
$\mathrm{sca}_{\check{\mathbb{H}}}$	0	4	0	2

	GS proofs		This work	
T	$\hat{\mathbb{G}}$	$\check{\mathbb{H}}$	$\hat{\mathbb{G}}$	$\check{\mathbb{H}}$
PPE	$4\lvert\hat{\mathbf{x}}\rvert + 8$	$2\lvert\check{\mathbf{y}}\rvert + 8$	$4\lvert\hat{\mathbf{x}}\rvert + 4$	$2\lvert\check{\mathbf{y}}\rvert + 4$
$\mathrm{PEnc}_{\hat{\mathbb{G}}}$	Undefined		$4\lvert\hat{\mathbf{x}}\rvert + 4$	$\lvert\check{\mathbf{y}}\rvert + 2$
$\mathrm{PConst}_{\hat{\mathbb{G}}}$	$2\lvert\hat{\mathbf{x}}\rvert$	0	$2\lvert\hat{\mathbf{x}}\rvert$	0
$\mathrm{PEnc}_{\check{\mathbb{H}}}$	Undefined		$2\lvert\hat{\mathbf{x}}\rvert + 2$	$2\lvert\check{\mathbf{y}}\rvert + 4$
$\mathrm{PConst}_{\check{\mathbb{H}}}$	0	$2\lvert\check{\mathbf{y}}\rvert$	0	$2\lvert\check{\mathbf{y}}\rvert$
$\mathrm{ME}_{\hat{\mathbb{G}}}$	$\lvert\hat{\mathbf{x}}\rvert + 4$	8	$\lvert\hat{\mathbf{x}}\rvert + 2$	4
$\mathrm{MEnc}_{\hat{\mathbb{G}}}$	Undefined		$\lvert\hat{\mathbf{x}}\rvert + 2$	2
$\mathrm{MConst}_{\hat{\mathbb{G}}}$	$\lvert\hat{\mathbf{x}}\rvert$	0	$\lvert\hat{\mathbf{x}}\rvert$	0
$\mathrm{MLin}_{\hat{\mathbb{G}}}$	0	0	0	0
$\mathrm{ME}_{\check{\mathbb{H}}}$	8	$\lvert\check{\mathbf{y}}\rvert + 4$	4	$\lvert\check{\mathbf{y}}\rvert + 2$
$\mathrm{MEnc}_{\check{\mathbb{H}}}$	Undefined		2	$\lvert\check{\mathbf{y}}\rvert + 2$
$\mathrm{MConst}_{\check{\mathbb{H}}}$	0	$\lvert\check{\mathbf{y}}\rvert$	0	$\lvert\check{\mathbf{y}}\rvert$
$\mathrm{MLin}_{\check{\mathbb{H}}}$	0	0	0	0
QE	4	4	2	2
$\mathrm{QConst}_{\hat{\mathbb{G}}}$	0	0	0	0
$\mathrm{QConst}_{\check{\mathbb{H}}}$	0	0	0	0

Figure 4.13: Scalar multiplications needed for computing commitments and NIZK proofs

so the prover can compute a commitment with 2 scalar multiplications instead of 4 scalar multiplications.

As shown in Fig. 4.13, by using the secret key sk the prover can reduce the number of scalar multiplications by 50% for commitments to group elements and commitments to elements in $\mathbb{Z}_p$. Computing NIZK proofs is more complicated and there are many operations that cannot be avoided by using the secret key sk. However, in some cases the improvement is very noticeable as in the case of quadratic equations ($T = \mathrm{QE}$) where the number of scalar multiplications is reduced by 50%[3]. Furthermore, in most applications found in the literature there are only a few variables in the equations, which makes our improvements more significant.

[3] We assume that operations in $\check{\mathbb{H}}$ are more computationally expensive than operations in $\hat{\mathbb{G}}$, as usually $\hat{\mathbb{G}}$ is an elliptic curve over a prime order field and $\check{\mathbb{H}}$ is the same elliptic curve over an extension field [GPS08]. Therefore, we have tried to reduce the numbers of operations in $\check{\mathbb{H}}$ as much as possible. In addition, we have for simplicity assumed that the commitments that appear in the NIZK proof have as many randomization factors as possible conditioned to the equation type T.

4.8.5 Security proofs for the prover-chosen CRS commit-and-prove scheme

Lemma 4.8.4. *The prover-chosen CRS commit-and-prove scheme has perfect correctness.*

Lemma 4.8.5. *The prover-chosen CRS commit-and-prove scheme has perfect soundness and perfect F-knowledge.*

Perfect correctness, perfect soundness and perfect F-knowledge can be proven in a similar way to how they are proven for our commit-and-prove scheme.

Lemma 4.8.6. *The prover-chosen CRS commit-and-prove scheme is composable zero-knowledge if the SXDH assumption holds relative to $\mathcal{PG}$.*

Proof. (Sketch) The zero-knowledge simulator is quite straightforward given a zero-knowledge simulator for our commit-and-prove scheme. First, the simulator will create the public key using the algorithm $\mathrm{SimProverGen}$ given in Fig. 4.11. Then, instead of making NIZK proofs to show that the generated public key is binding, it will simulate these proofs using the trapdoor key tk associated to ck. After simulating the well-formedness of pk, the simulator is able simulate any proof by using the secret key sk.

It should be noted that the simulator is simulating an NIZK proof for a false statement as the public key generated is no longer perfectly binding. In principle, the definition of zero-knowledge only guarantees that simulated proofs are indistinguishable from honestly generated proofs as long as the statement is true. However, in our case the public key and the simulated proof of well-formedness are computationally indistinguishable from a binding public key generated with the algorithm $\mathrm{ProverGen}$ and an honestly generated well-formedness proof.

To see that, consider the following hybrid proof: the first game corresponds to the setting where the public key is generated using $\mathrm{ProverGen}$ and the well-formedness proof is not simulated. The second game corresponds to the same setting as the first game, but the well-formedness proof is simulated. Both games are computationally indistinguishable due to the zero-knowledge property of Groth-Sahai proofs. In the third game, the public key is generated using $\mathrm{SimProverGen}$ and the well-formedness proof is simulated. The second game and the third game are computationally indistinguishable because the two types of public key are indistinguishable under the SXDH assumption. □

4.9 Applications of our results

In [BCKL08] the authors give a construction of a P-Signature scheme in asymmetric bilinear groups. This construction uses Groth-Sahai proofs based on the SXDH instantiation. The statements that have to be proven are such that our techniques can be applied to have better efficiency. Concretely, in the construction given in the paper each P-signature proof consists of 18 elements in $\hat{\mathbb{G}}$ and 16 elements in $\check{\mathbb{H}}$. Using our techniques, we can get 16 elements in $\hat{\mathbb{G}}$ and 16 elements in $\check{\mathbb{H}}$. In the full version of [BCKL08], they give another construction in which our techniques could be applied too, bringing the 12 elements in $\hat{\mathbb{G}}$ and 10 elements in $\check{\mathbb{H}}$ that they need to 10 elements in $\hat{\mathbb{G}}$ and 10 elements in $\check{\mathbb{H}}$. In addition, in the equations that appear in their scheme, the constant terms in $\mathbb{T}$ that appear are of the form $e(\hat{g}, \check{h})$. Using Groth-Sahai proofs, to get NIZK proofs they would need to add extra commitments and proofs. This is why they do not use NIZK proofs but NIWI proofs. However, using our techniques the same proofs are actually NIZK proofs, without increasing the number of elements needed. It would be interesting to see if this would result in more efficient P-signature schemes.

Another example is [AFG$^+$16], which defines structure-preserving signatures, i.e., signature schemes where the messages, signatures and verification keys are group elements and where the verification predicate is a conjunction of pairing-product equations. This definition makes structure-preserving signatures particularly well-suited for the application of Groth-Sahai proofs. One of their applications of structure-preserving signatures given in [AFG$^+$16] is a blind signature scheme in which Groth-Sahai proofs are used. To obtain a blind signature, the user commits to several elements and proves in zero-knowledge that some relations are satisfied. These are relations in which we can apply our techniques to get better efficiency. Concretely, the output of the Obtain protocol of the blind signature scheme consists of 18 elements in $\hat{\mathbb{G}}$ and 16 elements in $\check{\mathbb{H}}$, and using our techniques the output of the protocol can be reduced to 16 elements in $\hat{\mathbb{G}}$ and 14 elements in $\check{\mathbb{H}}$.

4.10 Conclusions

In this contribution we have presented several techniques to improve the efficiency of Groth-Sahai proofs. This work is based on the SXDH setting, but our techniques can be applied to other settings as well. Both framing Groth-Sahai proofs as a Commit-and-Prove scheme and letting the prover choose her own crs are contributions which clearly do not depend on the setting Groth-Sahai proofs are based on. Distinguishing base elements and the implied improvements can also be done in any Groth-Sahai proofs setting. Finally, although defining the encryption type can also be done for any setting, one has to be careful when using encryptions in Groth-Sahai proofs since we have to guarantee that we can still simulate those proofs.

In particular, the results from this contribution can also be applied to any settings which result from the assumptions given in Section 3.

Chapter 5

State of the Art in Electronic Voting

5.1 Types of electronic voting protocols

The main two requirements of an electronic voting system are privacy and verifiability. Historically, electronic voting systems were designed to meet the privacy requirements and then they were modified or enhanced to satisfy other requirements. For this reason, we classify electronic voting systems based on how they achieve privacy. We will follow the criteria used in [NBV14], where they classify the voting systems depending on in which phase of the election the relation between a voter and her vote is broken.

5.1.1 Code Voting

First, we consider schemes in which such relation is broken *before* the voting phase. There is currently only one family of such protocols, called *code voting* [Cha01b]. In code voting schemes, the election authorities generate a code card for each voter before the voting phase takes place. In each code card, there is a Code Card Identification Number, a Voting Code for each candidate and a Verification Code for each candidate (all codes vary from voter to voter). Once the code cards have been generated they are sent to each voter by a secure channel such as postal mail. In order to cast a vote, the voter sends the Code Card Identification Number and the Voting Code(s) corresponding to the chosen candidate(s) to the election authorities. The election authorities process these codes and return Verification Code(s) to the voter, who checks that they match the ones in her Code Card. At the end of the election, the election authorities use their knowledge of the relationship between voting codes and candidates to tally the election.

The main advantage of this class of electronic voting systems is that privacy does not depend on the voting device not being compromised. Indeed, the voting device does not get at any point information neither about the real identity of the voter nor about the selected candidates. This is why we say that privacy is broken before the voting phase.

However, this type of systems suffer from two important drawbacks. First, the code cards need to be sent to voters via some trustworthy channel such as postal mail. If this channel gets compromised then the attacker can break voter's privacy by relating the codes sent by the voter to her identity and to her selections. The attacker can also impersonate the voter, as it has all the information needed to cast a vote on the voter's behalf. The second drawback is usability: introducing codes is not as intuitive as using a graphical user interface to select the candidates and, if the number of candidates to be chosen is large, the amount of codes required makes the system unusable.

These systems are not infallible neither. Even though the voter does not introduce any personal information into the voting device, such device might extract some information about the behavior of the voter such as how long it takes for the voter to cast a vote. This information might help an attacker to create a profile of the voter. Therefore, it is not so clear that the voting device has no information about the voter's identity. Using this information, the voting device might make a guess of what voting option the voter might be choosing and carry out the following attack. The first time the voter attempts to vote, the voting device displays some error. Then, subsequent attempts using the same voting code (and, as a consequence, for the same candidate) are denied until the voter introduces another code. At this point the voting device allows the vote to be cast.

5.1.2 Two Agencies

There are two broad families of electronic voting systems that break the relation between voters and votes during the voting phase: *anonymous token*-based systems [OKNV12] and *blind signature*-based systems [FOO92]. Both systems are based on the principle of separation of duties, for this reason they are often known as Two Agencies systems.

Both families of systems work in a two-step process. In the first step, the voter authenticates to some authentication authority. After the authentication has taken place, the authentication authority provides the voter some information which allows her to cast a ballot to the election authority. The main characteristic of these systems is that the ballot that the voter sends does not contain any information about her identity. As long as the authentication process does not reveal the voters' selections and the two authorities are not corrupt at the same time, these systems guarantee privacy.

The difference between the *anonymous token*-based systems and the *blind signature*-based systems is in the authentication process. On the one hand, *anonymous token*-based systems are based on the voter and the authentication authority engaging in some computation which produces a token for the voter. The voter can use this token to anonymously send her voting option to the election authority, guaranteeing that she was authenticated with the authentication authority.

On the other hand, *blind signature*-based systems consist on having the voter blinding her selections and sending those blinded selections to the authentication authority. The authentication authority then authenticates the voter and produces a signature on the blinded selections, returning the signature to the voter. Finally, the voter can unblind the signed and blinded selections obtaining a signature produced by the authentication authority on her selections. The voter then sends her selections together with the signature to the election authority, who can check the signature to verify that the voter was authenticated by the authentication authority.

This family of systems bases the privacy guarantees on the strong implicit assumption that the voting channel is anonymous. If the channel used is the Internet, there are many ways an attacker could de-anonymize the voter, from IP addresses to her browser user agent [Eck10].

5.1.3 Homomorphic Tallying

We now move to the first family of schemes which break the relation between votes and voters *after* the voting phase. These schemes, first introduced in [CGS97], work as follows. During the voting phase, the voting client encrypts the selected voting option using the electoral board public key. The encrypted vote is then sent to the electoral board, who stores all the encrypted votes until the end of the election. At the end of the election, the electoral board *aggregates* all the encrypted votes obtaining an encryption of the tally. Finally, the electoral board decrypts the aggregation of the votes obtaining the tally. To be able to aggregate the encrypted votes and obtain the encryption of the tally, the encryption scheme must be homomorphic.

More in detail, in order to cast a vote the voting device proceeds as follows. Using the Pailler encryption scheme as an example, for each eligible candidate the voting device encrypts either 1 if the candidate was chosen or 0 if the candidate was not chosen, obtaining a ciphertext for each eligible candidate. When tallying the election, the electoral board multiplies all the ciphertexts corresponding to the same candidate, obtaining an encryption of the number of people who chose this candidate. Finally, the electoral board decrypts each resulting ciphertext to obtain, for each candidate, the number of votes that she received.

In this system, the ballot cast by the voting device does not need to be anonymous. Indeed, in order to guarantee that only eligible voters participate in the election, voters sign their ballots with their private signing key. It is the homomorphic aggregation what breaks the relation between each voter and her chosen voting option: this is why we say that such relation is broken *after* the voting phase.

The description of the system given above is not complete though. As currently described, instead of encrypting values 0 or 1 for each candidate the voting device could encrypt any value such as 1000 or -1000. This would have the effect of voting 1000 times for the same candidate or subtracting 1000 votes from the votes received by the candidate. Similarly, with the system described above there is no way to restrict the number of candidates that a voter can choose.

The solution to this problem is to use zero-knowledge proofs to force the voting device to encrypt only values 0 or 1 and to choose up to the allowed number of candidates. There exist specific zero-knowledge proofs which are efficient enough so that the process of casting a ballot can still be completed in a reasonable amount of time.

The main drawback of this type of schemes is that one encryption has to be done for each candidate.

When the number of eligible candidates is small, such as in a referendum or small elections, this does not suppose any problem. However, in big elections with tens of parties and hundreds of eligible candidates per party the number of encryptions to be done is too big to be completed in a reasonable time.

A partial solution to this problem is to encrypt more than one candidate per ciphertext in a way that the ciphertexts can still be aggregated obtaining the vote count for each candidate. This can only be done with special encryption schemes such as the Damgaard Jurik encryption scheme [DJ01]. However, this special encoding of the voting option imposes some restrictions on the number of eligible candidates and the number of voters. For large elections, this approach is not the most efficient from the voters' side.

5.1.4 Mix-Nets

The second family of voting schemes which break the relation between votes and voters *after* the voting phase are Mix-Net-based voting schemes, first introduced in [SK95]. In such schemes, the voting device encrypts the selected candidates, similarly to what is done in homomorphic tallying-based voting schemes. The electoral board receives all ballots and stores them until the end of the election. At the end of the election, the electoral board removes any voter-related information of the vote (such as a digital signature), shuffles all the ballots and decrypts them individually. Shuffling the ballots breaks any relation between votes and voters.

Shuffling the ballots instead of aggregating them has two important consequences. First, the encryption scheme being used does not need to have any special homomorphic property. Second, as no homomorphic aggregation of the candidates is performed, the voting client only has to encrypt the candidates chosen by the voter instead of encrypting all the eligible candidates. In some cases, all chosen candidates can even be encoded in a single plaintext so that the voting device only needs to perform one encryption. This implies that the voting process is cheap in computational terms.

In contrast with homomorphic tallying-based systems, in Mix-Net based systems voters do not need to compute any Zero-Knowledge proof to show that the ciphertext does not contain invalid options. However, this has no effect on the tally of the election. Since the election authorities decrypt each ballot individually, invalid votes can be identified and discarded.

The main drawback of this scheme is that the election authorities need to perform the shuffling of the votes and to decrypt each individual vote. Compared to homomorphic tallying-based systems, this requires a higher computational power from the election authorities. As we will see later, expensive zero-knowledge proofs are added in the tallying process to make such process verifiable. In this case, the tallying process becomes even more computationally expensive than homomorphic tallying-based systems.

5.2 Verifiable electronic voting systems

We will now explain what are the main techniques which are used in order to provide verifiability mechanisms to electronic voting schemes. We will focus on homomorphic tallying-based and Mix-Net-based voting schemes, the other families of schemes suffer from very important drawbacks as we have explained.

Verifiability properties can be divided in four types, depending on which is the object of the verifiability. These four types are: cast-as-intended verifiability, recorded-as-cast verifiability, eligibility verifiability and counted-as-recorded verifiability. In this section, we will explain how these verifiability properties are usually achieved.

5.2.1 Cast-as-intended verifiability

We say that a voting system has cast-as-intended verifiability if the voter can verify that the ballot cast by the voting device contains her chosen candidates. In other words, the voting system should provide mechanisms so that the voter can check that the voting device did not cheat by encrypting candidates other than her selections. There are three main mechanisms for providing cast-as-intended verifiability: *immediate decryption*, *interactive proofs* and *return codes*.

The **immediate decryption** mechanism, introduced in [Ben06], works for both homomorphic tallying-based and Mix-Net-based systems. This mechanism works as follows. First, the voting device produces the ballot by encrypting the chosen candidates. Then, the voting device commits to this ballot either by printing it or showing it on screen. Once the voter has access to the ballot, she might choose to audit the ballot or to cast it. If she chooses to audit the ballot, then the voting device provides information which allows the voter to perform the cast-as-intended verification. Such information is usually the randomness used in the encryption process. The voter must then use an independent trusted device to verify that her ballot contains her selected candidates.

However, this information provided by the voting device would also serve as a receipt which could be used for coercion or vote selling. Therefore, if the voter chooses to audit the ballot then this ballot can not be cast and the voting process has to be started from the beginning. The voting device then produces another ballot, which the voter can choose to audit or to cast it.

The main drawback of this mechanism is usability: it might be hard for voters to understand the process. This, in turn, implies that many voters might choose not to audit their ballots and, in conclusion, the effectiveness of this mechanism might be reduced. In case that each voter uses a distinct voting device, this implies that the mechanism is effective for a given voter only if she uses the mechanism.

The **interactive proof** mechanism, introduced in [Ben06], also works for homomorphic tallying-based and Mix-Net-based voting systems. In this mechanism, voters and voting devices engage in an elaborate interactive protocol to convince the voter that the ballot contains her selections. First, the voting device produces several ballots containing the candidates chosen by the voter. The voting device commits to those ballots by printing them in a location or form that can not be read by the voter (such as printing them behind an opaque screen or on the back of a card). Then, the voter chooses a subset of those ballots randomly and the voting device shows, similarly to as done in the immediate decryption mechanism, that the ballots correspond to the candidates chosen by the voter. Finally, the voting device proves in zero-knowledge that all the ballots correspond to the same voting option (without revealing which one it is) and produces *fake* audited ballots corresponding to voting options not chosen by the voter and simulated zero-knowledge proofs for these fake ballots.

This mechanism suffers from several drawbacks. First, it needs to print the ballots in such a way that the voter does not see them. This requires special technology not available to every voter. In addition, voters must make several random selections. As humans, we are bad at choosing randomness and, as a result, the selections would not be purely random and the effectiveness of the system would decrease. Finally, the usability of this system is, at best, questionable.

There is another mechanism to provide cast-as-intended verifiability, which are the so-called **return codes** [Gjø11]. As far as we know, the current proposals for the return codes mechanism only work with Mix-Net-based electronic voting systems. The mechanism borrows an idea from *Code Voting* systems: once the voter has submitted her ballot, the election authorities compute some *Verification Codes* (also known as *Return Codes*) from the ballot and send such codes to the voter. The voter then checks that these codes match some other codes which appear in her *Code Card*, which the voter received by postal mail prior to the election. If the codes match, this guarantees that the ballot contains the voting option chosen by the voter.

Computing the *Verification Codes* from the ballot without breaking ballot privacy is a challenging task. Current proposals achieve this goal by using homomorphic encryption schemes. In [Gjø11], the computation of the *Verification Codes* is split in two entities, each of them doing complementary computations, and ballot privacy is guaranteed as long as at least one of the entities is honest. On the other hand, in [GGP15] the ballot produced by the voting device consists of two parts: the encryption of the voting option and the *Partial Verification Codes*. The voting device also provides some zero-knowledge proofs to guarantee that the *Partial Verification Codes* correspond to the encrypted voting option. The election authorities use the *Partial Verification Codes* to compute the actual *Verification Codes*.

This is clearly the most usable cast-as-intended verification mechanism of the three that we exposed. Indeed, the voter only needs to check that the codes print on screen match with the corresponding codes on the Code Card. However, this mechanism has two drawbacks. First, as opposed to the other mechanisms, it is necessary to deliver the Code Cards by postal mail. This increases the cost of the election and might not be feasible in all scenarios. In addition, the mechanism is effective as long as voters do check that the codes match. This is similar to what happens with the other two mechanisms of cast-as-intended verifiability.

5.2.2 Recorded-as-cast verifiability

A system with recorded-as-cast verifiability allows the voter to verify that the ballot cast by her voting device was correctly received and stored by the election authorities. In particular, the system provides evidence to convince the voter that the ballot was not modified nor deleted after it was cast.

There is currently only one mechanism which provides recorded-as-cast verifiability, known as the *Bulletin Board* [GV10]. The idea is quite simple: during and/or after the voting period, the election authorities publish all the ballots cast by the voters to some repository, for example a website. When a ballot is cast, the voting device also prints the ballot (e.g., on a screen). The voter can then check that the ballot printed by the voting device also appears on the repository.

Although the idea is simple, there are some real-world implementation subtleties. For example, there have to be mechanisms to verify that the same Bulletin Board is shown to all the voters, otherwise the Bulletin Board would lose its effectiveness. Similarly, if the Bulletin Board is published during the voting period, it has to be guaranteed that the Bulletin Board's contents are updated in an append-only way. Despite these subtleties, the Bulletin Board is the only recorded-as-cast verifiability mechanism known so far and recorded-as-cast verifiability is considered a solved problem.

5.2.3 Eligibility verifiability

In addition to allowing voters to verify that their vote was correctly stored by the system, an electronic voting system should also allow anyone to verify that the ballot box only contains ballots cast by voters who can participate in the election. This property is known as eligibility verifiability.

Eligibility verifiability is easily accomplished by digitally signing each ballot with a voter's private key. This assumes that voters have key pairs which public key is recognized by the election authorities. This is easily achieved if voters have some electronic id card (such as Spanish id card [edn]), which usually contain at least one signature key pair.

It should be noted that, by signing each ballot, anyone can verify whether a voter participated or not in an election. This is a concern when strong notions of privacy are required in which individual participation data should also be private. We will not consider these strong notions of privacy in this thesis.

5.2.4 Counted-as-recorded verifiability

Finally, a voting system should also give evidence that the tallying process was done correctly, taking as input the ballots stored in the ballot box. A system providing such evidence is said to have counted-as-recorded verifiability.

The mechanisms for providing counted-as-recorded verifiability depend on the family of the electronic voting system. Once again, we will only focus on homomorphic tallying-based voting systems and Mix-

Net-based voting systems. Although the mechanisms for these schemes are different, the core idea is the same: to provide a proof that all secret computations were done correctly.

When considering **homomorphic tallying**-based systems, the mechanism for achieving counted-as-recorded verifiability is relatively simple. It can be split in two parts. First, the homomorphic aggregation of the ciphertexts is a deterministic, non-private operation, therefore it can be reproduced and no proofs need to be given for this operation. The decryption operation is a private operation, so zero-knowledge proofs are given to show that the ballots were correctly decrypted. An example of such proofs can be found in [CGS97].

On the other hand, counted-as-recorded verifiability mechanisms for **Mix-Net**-based systems are a bit more complex. The reason is that both the shuffling and the decryption processes are private. In addition, when shuffling the ballots the output of the shuffle needs to be kept private since the shuffle could be inferred otherwise. The most common solution is to shuffle and re-encrypt each ballot and then decrypt each individual ballot. As we already mentioned, decryption zero-knowledge proofs are fairly simple. On the other hand, Zero-Knowledge proofs for the shuffle and re-encryption process are more complex. The most efficient ones are those given in [BG12] and [TW10].

5.3 Going formal: syntactical and security definitions

5.3.1 Syntactical definition

Once we have given an overview of the current families of electronic voting schemes, we will explain what work has been done on formal security definitions. The focus is going to be on homomorphic tallying-based and Mix-Net-based voting systems. We will also restrict this section to privacy definitions since previous verifiability definitions are not relevant for this thesis. We will give the privacy definitions given in [BCG$^+$15], since they are the most recent ones and they fix some flaws found in previous definitions.

We will first give the syntactical definition of an electronic voting system. This specifies the processes which take place in the system together with their inputs and outputs. As explained in Section 5.1, in a homomorphic tallying-based or in a Mix-Net-based electronic voting system the voters only interact with the system by casting their ballot. These type of schemes are said to be single-pass voting schemes, as defined in [BPW12b].

Before introducing the syntactical definition, it will be useful to define the entities that participate in an election. First, *election authorities* are in charge of defining the election parameters, generating any required cryptographic keys and tallying the result of the election. The *bulletin board* (BB) is a repository of information containing public keys and ballots. It can be read by any entity but only the *bulletin board manager* and the election authorities can write to it. *Voters* participate in the election by choosing their preferred voting option and submitting their ballots.

In addition, there are some parameters which define a single-pass voting scheme. These are the set of possible votes $\mathbb{V}$ (i.e., all valid combinations of candidates), a result space R and a result function $\rho : (\mathbb{V} \cup \{\perp\})^* \to R$, where $\perp$ denotes an invalid vote.

The result function usually specifies the rules applied to the votes for obtaining the tally. However, the output of the result function is not necessarily the ultimate result of the election. Indeed, in some cases it might just specify some process after which the count of the votes is performed in a public way. The clearest example of such result function is the *multiset function*. As defined in [BCG$^+$15], the multiset function takes as input all the cast votes and outputs them in a random order. Once the votes have been disclosed in a random order any entity can compute the result of the election in a public way.

We can now give the syntactical definition of a single-pass voting scheme. A single-pass voting scheme is defined by the following (possibly probabilistic) algorithms:

$\mathsf{ElectionSetup}(1^\lambda)$: on input a security parameter 1^λ it generates and outputs an election public key pk and an election secret key sk. This algorithm defines all keys and parameters which will be needed in the electronic voting system. It might include keys for encryption and signature schemes, but also common reference strings for NIZK proofs.

$\mathsf{Vote}(pk, v)$: on input the election public key pk and a vote $v \in \mathbb{V}$, outputs a ballot b.

$\mathsf{ValidateBallot}(BB, b)$: on input a bulletin board BB and a ballot b it outputs either accept (1) or reject (0).

$\mathsf{Tally}(sk, BB)$: on input the election secret key sk and the bulletin board BB it outputs the tally $r \in R$ together with a proof of correct tabulation Π.

$\mathsf{VerifyTally}(BB, r, \Pi)$: on input the bulletin board BB, the tally r and a proof of correct tabulation Π it outputs either accept (1) or reject (0).

Now we explain which algorithms are executed by which entities. A single-pass protocol is executed in the following three phases.

1. In the *setup* phase, the election authority runs the $\mathsf{ElectionSetup}$ algorithm. It publishes the election public key pk to the bulletin board BB and privately keeps the election secret key sk.

2. In the *voting* phase, the voter chooses a vote v and creates a ballot using the Vote algorithm, where the public key pk is retrieved from the bulletin board. The result of the Vote algorithm is a ballot b, which is sent to the bulletin board manager. The bulletin board manager then executes the $\mathsf{ValidateBallot}$ algorithm on the ballot. If the algorithm returns 1, then the bulletin board manager adds the ballot to the bulletin board. Otherwise, it rejects the ballot and notifies the voter.

3. In the *counting* phase, the election authority runs the Tally algorithm on the bulletin board using the election secret key. The output of the Tally algorithm, which consists of the result and the correct

tabulation proof, is published to the bulletin board. This proof of correct tabulation can then be verified by any entity using the $\mathsf{VerifyTally}$ algorithm.

5.3.2 Correctness

Before considering privacy definitions, we have to define a basic requirement: if the voting system is used as intended then it should behave in a pre-defined manner. In particular, we want that if all participants are honest then (i) the result r output by the Tally algorithm is equal to the evaluation of the result function ρ on the voting options corresponding to the ballots cast by the voters and (ii) the algorithm $\mathsf{VerifyTally}$ on input the result of the Tally algorithm returns accept.

5.3.3 Privacy

Defining privacy for a voting scheme is a surprisingly difficult task. One of the reasons is that the access of an adversary is quite strong: on the one hand, it has access to the result of the election and, on the other hand, some voters and some members of the election authority might cooperate with the attacker. We say that those voters or election authorities are corrupt.

If not all the members of the election authority are corrupt, then there are techniques which can help to make the voting system private. These tools are known as Multi-Party Computation [Yao82] and can be implemented independently of the other parts of the voting system. For sake of simplicity we will assume that there is only one election authority, which is honest.

As we said, the adversary might corrupt some voters as well. We might attempt to define privacy as no adversary being able to tell what honest voters voted for. In the extreme situation where the adversary corrupts all-but-one voters then the adversary might force a draw with the ballots of all corrupt voters and the tally would reveal the honest voter's vote. Still, if the adversary does not corrupt all voters, there are other situations in which it could trivially know the honest voters' choices. For example, if the adversary controls one half minus one of the voters it could submit ballots for the same candidate. This would reveal if all other voters voted for the opposing candidate or not.

A privacy definition should allow an adversary to get such information as such leakage is unavoidable when the adversary can corrupt voters. However, it should not allow any other information leak. This is captured in the Universal Composability (UC) model [Gro04]. This model works by specifying an ideal functionality, in our case having all voters submit votes privately to a trusted party and this trusted party publicly announcing the tally. Some of the voters could be corrupt, so their votes would be leaked. Then, a scheme satisfies privacy if it does not leak any information not leaked in the ideal functionality.

Working in the UC model is extremely complex. However, in [BCG$^+$15] a definition is given which is proven to be equivalent to the UC definition. Their definition is not intuitive, but it is easy to work with. In addition, as it is equivalent to the UC definition we can be sure that it is a definition which matches our expectations as defined in the UC definition.

The definition given in [BCG$^+$15] works as follows. The adversary might corrupt voters, submitting ballots on their behalf. In addition, for each honest voter it submits two possible votes. Either the first vote will be used to cast ballots for all voters or the second vote will be the one used to cast ballots. The adversary will then have to distinguish whether the first vote or the second vote was used to cast each voter's ballot. Up to this point this is a typical indistinguishability definition where the adversary must distinguish between two situations, or experiments, which have similar behavior.

To avoid having the tally leak information about the honest voters' votes, as in the case of all-but-one voters being corrupt, the same tally will always be shown to the adversary, regardless of whether the first vote or the second vote was used to cast honest voters' ballots. This implies that the tally the adversary gets might not correspond to the votes cast by honest and corrupt voters. However, this allows us to have schemes in which the adversary does not distinguish between the two scenarios.

This is formalized by considering two experiments or situations: the left (L) experiment and the right (R) experiment. In both experiments, there is a bulletin board (BB_L and BB_R respectively). The adversary might submit ballots b, which will be added to both bulletin boards if the ballots are valid. In addition, the adversary might submit two votes for each honest voter. These votes correspond to the right vote, or the vote to be used in the right experiment, and the left vote, which will be used in the left experiment. At the end of the election, the election authorities tally the ballots on the bulletin board but always output the result of the left experiment.

There is one caveat here: how can the election authorities give a proof of correct tabulation if the announced result is not the tally of the election? In this case, the election authorities need to provide a *simulated* proof of correct tabulation. The security definition must allow the election authority to provide such simulated proofs. Note how a security definition is allowed to deviate from the usual work flow: in a real election, it must be impossible to have the election authorities provide a simulated proof of correct tabulation.

Another minor caveat is that the proof of correct tabulation and the simulated proof of correct tabulation could be used as covert channel. This was studied in [BCG$^+$15] and it was shown that previous privacy definitions had a flaw that allowed a voting scheme which was clearly not private to satisfy the privacy definition by using the covert channel. This flaw was fixed in the definition of [BCG$^+$15] by restricting the information from which the simulated proof of correct tabulation was computed.

Ballot privacy is defined by using two experiments between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The goal of the adversary is to distinguish between the two experiments. For compactness, we present the two experiments as a single experiment which depends on a bit $\beta \in \{0, 1\}$. Both experiments assume given the set of voting options $\mathbb{V}$, the result space R, the result function ρ and use an algorithm $\mathsf{SimProof}(BB, r)$ which, given a bulletin board and a result, simulates a correct tabulation proof (in this case, the $\mathsf{ElectionSetup}$

algorithm will need to be modified so that proofs can be simulated).

The experiment Exp_β is run in these phases:

1. **Setup phase**. $\mathcal{C}$ sets up two empty bulletin boards BB_L and BB_R. It runs the $\mathsf{ElectionSetup}(1^\lambda)$ protocol to obtain the election public key pk and the election private key sk. It then posts pk on both bulletin boards. The adversary is given read access to either BB_L if $\beta = 0$ or BB_R if $\beta = 1$.

2. **Voting phase**. The adversary may make two types of queries:

   - **Vote**(v_L, v_R) queries. The adversary provides two votes $v_L, v_R \in \mathbb{V}$. The challenger runs $\mathsf{Vote}(pk, v_L)$ and $\mathsf{Vote}(pk, v_R)$ obtaining two ballots b_L and b_R respectively. $\mathcal{C}$ then obtains new versions of the boards BB_L and BB_R by running both $\mathsf{ValidateBallot}(BB_L, b_L)$ and $\mathsf{ValidateBallot}(BB_R, b_R)$ and updating the boards accordingly.
   - **Ballot**(b) queries. These are queries made on behalf of corrupt voters. Here the adversary provides a ballot b. The challenger runs $\mathsf{ValidateBallot}(BB_L, b)$. If the algorithm returns 1, BB_L is updated and $\mathsf{ValidateBallot}(BB_R, b)$ is executed, updating BB_R accordingly. Otherwise, if the algorithm returns 0, it does nothing.

3. **Tallying phase**. The challenger evaluates $\mathsf{Tally}(sk, BB_L)$ obtaining the result r and the proof of correct tabulation Π. If $\beta = 0$, the challenger posts (r, Π) on the bulletin board BB_L. If $\beta = 1$, the challenger runs $\mathsf{SimProof}(BB_R, r)$ obtaining a simulated proof Π' and posts (r, Π') on the bulletin board BB_R.

4. **Output**. The adversary $\mathcal{A}$ outputs a bit α.

We say that a voting protocol for $(\mathbb{V}, R, \rho)$ provides ballot privacy if there exists an algorithm $\mathsf{SimProof}$ such that for any p.p.t. adversary $\mathcal{A}$ the following advantage is negligible in the security parameter λ.

$$\mathbf{Adv}^{\mathsf{priv}}_{\mathsf{SimProof}}(\lambda) := |\Pr[\alpha = 1|\beta = 1] - \Pr[\alpha = 1|\beta = 0]|$$

5.3.4 Strong consistency and strong correctness

The definition of privacy given above does not suffice to be equivalent to the UC definition. As shown in [BCG$^+$15], two extra properties are needed: strong consistency and strong correctness.

While correctness states that when all participants are honest then the output of the tally must match the output of the result function on the votes chosen by the voters, it does not specify the behavior of the system when some voters are not honest. In particular, it could happen that a corrupt voter caused the tally of the election to be completely different than the result function applied to the votes chosen by voters. Similarly, it could happen that corrupt voters submitted ballots that caused ballots cast by honest voters to be invalid.

Strong consistency states that the tally of the bulletin board must correspond to the result of applying the result function to the *contents* of the ballots in the bulletin board. This must hold even if the bulletin board was generated adversarially.

Strong consistency is given by the following game, parameterized by the election parameters $(\mathbb{V}, R, \rho)$ and an algorithm $\mathsf{Extract}(sk, b)$, which takes as input the election secret key and a ballot and outputs either a vote or the error symbol $\perp$ denoting an invalid vote.

1. **Setup phase**. The challenger runs $\mathsf{ElectionSetup}(1^\lambda)$ to obtain the election public key pk and the election secret key sk. It gives both pk and sk to the adversary $\mathcal{A}$.

2. **Bulletin Board**(BB). The adversary submits a bulletin board BB to the challenger.

3. **Output**. The challenger runs $\mathsf{Tally}(sk, BB)$ to obtain a result r and a correct tabulation proof Π. The output of the game is a bit γ. This bit is defined as 1 if $r \neq \rho(\mathsf{Extract}(sk, BB))$ and 0 otherwise, where $\mathsf{Extract}$ is applied on the bulletin board by applying it to each individual ballot.

We say that a voting protocol for $(\mathbb{V}, R, \rho)$ has strong consistency with respect to an extract algorithm $\mathsf{Extract}$ if the following conditions are satisfied:

(i) For any (pk, sk) generated by $\mathsf{ElectionSetup}$, for any vote $v \in \mathbb{V}$ we have $\mathsf{Extract}(\mathsf{Vote}(pk, v)) = v$

(ii) For any p.p.t. adversary $\mathcal{A}$, the following advantage is negligible in the security parameter λ:

$$\mathbf{Adv}^{\mathsf{s\text{-}const}}(\lambda) := \Pr[\gamma = 1]$$

On the other hand, strong correctness states that ballots created by honest voters should be valid regardless of the ballots submitted by corrupt voters. Strong correctness is given by the following game:

1. **Setup phase**. The challenger runs $\mathsf{ElectionSetup}(1^\lambda)$ to obtain the election public key pk and the election secret key sk. It gives both pk and sk to the adversary $\mathcal{A}$.

2. **Bulletin Board**(BB). The adversary submits a bulletin board BB to the challenger.

3. **Vote**(v). The adversary submits a vote v to the challenger.

4. **Output**. The challenger runs $\mathsf{Vote}(pk, v)$ to obtain a ballot b. The output of the game is a bit δ. This bit is defined as 1 if $\mathsf{ValidateBallot}(BB, b) = 0$ and 0 otherwise.

We say that a voting protocol for $(\mathbb{V}, R, \rho)$ has strong correctness if for any p.p.t. adversary $\mathcal{A}$, the following advantage is negligible in the security parameter λ:

$$\mathbf{Adv}^{\mathsf{s\text{-}corr}}(\lambda) := \Pr[\delta = 1]$$

5.4 Enc2Vote: a very basic voting scheme

In [BCP$^+$11], a very basic voting scheme is introduced. This scheme, called Enc2Vote (encrypt to vote), has as its only goal to satisfy ballot privacy, strong correctness and strong consistency. In particular, no verifiability requirements are satisfied. The idea of this voting scheme is to construct a very basic ballot private scheme which can be used as a basis for other, more advanced voting schemes.

The scheme is a Mix-Net based voting scheme and, in particular, uses as its result function ρ the multiset function. It uses as building block an encryption scheme (Setup,KeyGen,Enc,Dec) and works as follows:

ElectionSetup(1^λ): executes the Setup and KeyGen algorithms of the encryption scheme to obtain a setup gk and a pair of encryption keys (pk_e, sk_e). It sets $pk = (gk, pk_e)$ as the election public key, $sk = sk_e$ as the election private key and the message space of the voting scheme $\mathfrak{M}_{gk}$ as the set of votes.

Vote(pk, v): on input the encryption public key $pk = (gk, pk_e)$ and a vote $v \in \mathfrak{M}_{gk}$ it computes the ballot $b = \mathsf{Enc}(gk, pk_e, v)$.

ValidateBallot(BB, b): on input a bulletin board BB and a ballot b outputs 0 if $b \in BB$, and 1 otherwise.

Tally(sk, BB): on input the encryption secret key $sk = sk_e$ and the bulletin board BB, retrieves $pk = (gk, pk_e)$ from the bulletin board, decrypts all ballots and outputs $\rho(\{\mathsf{Dec}(gk, sk_e, b)\}_{b \in BB})$ and no auxiliary data.

VerifyTally(BB, r, Π): on input a bulletin board BB, the result r and a proof of correct tabulation Π always returns 1.

As shown in [BPW12b], the Enc2Vote scheme is ballot private, strongly correct and strongly consistent if the encryption scheme is NM-CPA. As we will see, our voting schemes will reduce some of their security properties to the security of the Enc2Vote scheme.

Chapter 6

Vote Validatability in Mix-Net-Based Electronic Voting

This section is based on the results published in [BEM15], coauthored with Pedro Bibiloni and Paz Morillo.

6.1 Motivation

When building a Homomorphic Tallying based voting system, a technical requisite is that the voter must construct a proof that her vote conforms to the election rules. Otherwise, the homomorphic aggregation of invalid votes could produce completely unreasonable results.

Mix-Net-based systems do not have this requirement. As votes are individually decrypted, it can be checked whether each decrypted vote conforms to the election rules and, in case it does not, consider it an invalid vote. From a technical perspective this is completely reasonable, we do not need to ask voters for a proof of her vote conforming to the election rules in order to have a secure Mix-Net-based system.

Despite proofs of ballot well-formedness not being necessary to implement a secure electronic voting system, the lack of such proofs might affect the reputation of the system. From a non-technical voter's perspective, it is reasonable to assume that if the voting interface does not allow for an invalid vote to be cast then invalid votes should be impossible to cast. Therefore, modifying the voting client to cast an invalid vote might be seen as an attack against the system, *even if it has no effect on the result on the election*. Besides, it would be impossible to track the attacker due to the anonymity provided by the Mix-Net. In addition, should there be a software bug which created ballots for invalid votes inadvertently, this would be only detected at the tallying phase. Depending on the number of invalid votes, the election might even have to be restarted – with the reputation loss that it represents.

6.2 Our contributions

In the paper [BEM15], we introduced the concept of *vote validatability*, which attempts to solve the problem mentioned above. We consider that an electronic voting system has vote validatability if it can be *publicly verified* that a ballot contains a vote conforming to the election rules – we want to be able to detect whether a vote is invalid before it is decrypted. This means that (a) no invalid votes will appear during the tallying of the election and (b) any software bug in the voting devices affecting ballot creation will be detected during the election period, so it can be quickly fixed, providing the voters another attempt to vote before the end of the election.[1]

As we discussed above, Homomorphic Tallying systems have vote validatability since it is a requisite in order to have a secure voting system, in contrast with Mix-Net-based systems. Adding this property to a Mix-Net-based scheme is a problem solved from a theoretical point of view: there are inefficient cryptographic tools such as general-purpose zero-knowledge proofs which can be used to achieve it. The challenge is thus using appropriate protocols to have a low computational cost from the voters' side, which is one of the advantages of using a Mix-Net-based system.

There is a trivial approach to achieve vote validatability: considering all possible contents of a ballot as valid. This can be done by defining an encoding for all-but-one eligible candidates and assigning any other encoding to the last candidate. However, this has some drawbacks. First, having several encodings for the same candidate opens the door to facilitating vote selling by making it possible to introduce the voter's identity in the encoding of the candidate. In addition, requiring a specific encoding might prevent benefiting from some features of the electronic voting systems. One example is the so-called return codes [Gjø10], which current implementations require that each candidate has only one encoding. Another one is using special encodings to aggregate encryptions before the tallying, as also done in [Gjø10]. Therefore, we prefer a solution to vote validatability which does not require a specific encoding of the candidates.

The rest of this chapter is structured as follows. We introduce a formal definition for the concept of vote validatability in Section 6.3. Then, we give a general construction of a Mix-Net-based scheme achieving vote validatability and ballot privacy. This construction is based on basic cryptographic primitives and is given in Section 6.4, along with the proofs of its security properties. Finally, we give a concrete, efficient instantiation of a Mix-Net-based system with vote validatability in Section 6.5.

6.3 Vote validatability definition

We now present the definition of vote validatability, which is the first contribution of the paper [BEM15]. Our definition of vote validatability applies to single-pass voting schemes, which syntactical definition is

[1] There could be bugs in the software which verifies vote validatability. However, this verification can be done in parallel by different implementations done by different entities, leveraging this risk.

given in Section 5.3.1. Vote validatability states that a ballot which passes all validations *must* correspond to a valid vote. This is modeled by stating that there must exist an algorithm $\mathsf{Extract}$ which never returns the error symbol $\perp$ on ballots for which $\mathsf{ValidateBallot}$ returns 1. We will also need that the voting protocol must be strong consistent with respect to $\mathsf{Extract}$.

Vote validatability is given by the following game parametrized by the election parameters $(\mathbb{V}, R, \rho)$ and an algorithm $\mathsf{Extract}(sk, b)$, which takes the election secret key and a ballot and outputs either a vote or the error symbol $\perp$ denoting an invalid vote.

1. **Setup phase**. The challenger runs $\mathsf{ElectionSetup}(1^\lambda)$ to obtain the election public and private keys (pk, sk), giving both to the adversary.
2. **Ballot**(b). The adversary submits a ballot b to the challenger.
3. **Output**. The output of the game is a bit δ. This bit is defined as 1 if $\mathsf{Extract}(sk, b) = \perp$ and $\mathsf{ValidateBallot} = 1$, and as 0 otherwise.

We say that a single-pass voting protocol for $(\mathbb{V}, R, \rho)$, as defined in Section 5.3.1, has vote validatability with respect to an extract algorithm $\mathsf{Extract}$ if the following conditions are satisfied:

(i) The voting protocol for $(\mathbb{V}, R, \rho)$ is strongly consistent (see Section 5.3.4) with respect to $\mathsf{Extract}$

(ii) For any p.p.t. adversary $\mathcal{A}$, the following advantage is negligible in the security parameter λ:

$$\mathbf{Adv}^{\mathsf{val}}(\lambda) := \Pr[\delta = 1]$$

One implication of the definition given above is that, if the protocol has vote validatability, then it must be satisfied that, for any honestly-generated keys and any adversarially generated bulletin board, the result output by the tally can be obtained with only valid votes, $r \in \rho(\mathbb{V}^*)$.

We want to remark that vote validatability does not depend on the secrecy of the election secret key. However, it assumes that the $\mathsf{ElectionSetup}$ is run honestly. Even though this can be achieved by distributing the trust among multiple authorities, we give the definition assuming that there is only one authority for the sake of simplicity.

6.4 General construction

6.4.1 Core idea

In an electronic voting system, voters might be able to vote for more than one candidate, so we will consider a generic scenario in which votes are subsets of n distinct candidates from a larger but specified list of them. Treating the set of votes as the set of combinations of candidates would result in a terribly inefficient system. Therefore, each of the selected candidates will be encrypted independently. To prove that each candidate hidden in its respective encryption belongs to the list of candidates, we will use a set membership protocol based on digital signatures. In addition, we will use a technique originally designed to avoid double-spending in e-cash to demonstrate that the candidates hidden in these encryptions are distinct.

The main idea of our new construction is inspired by the set membership protocol proposed by Camenisch et al. [CCas08]. In that work, the authors construct a protocol for proving that a value is a commitment to a member of a pre-defined set. Their protocol works as follows. First, there is a trusted third party which produces signatures on each element of the set. Then, the prover constructs a zero-knowledge proof that she knows a signature on the committed value which verifies under the trusted third party's secret key. When the encryption scheme and the signature scheme being used have *nice* structural properties, the size of the proof is small and constant on the size of the set. In our case, the electoral authority will sign all candidates, and the voter will prove that she knows a signature on each selected candidate. However, this would still allow the voter to choose repeated candidates.

To detect this last situation, we use a technique inspired by the e-cash scheme given in [BCKL09]. In e-cash, detecting double-spending is essential, and this problem is similar to detecting repeated candidates in a vote. We will ask the voter to choose a pseudo-random permutation key and to publish the image of each chosen candidate under the pseudo-random permutation defined by such key. Given that the pseudo-random permutation is deterministic, if the voter chooses the same candidate more than once this will be detected by any entity. The protocol will require the voter to prove in zero-knowledge that the images of the pseudo-random permutation correspond to the candidates which she encrypted and that she knows a signature for each candidate.

In Section 6.4.2 we describe a generic protocol built on the mentioned cryptographic primitives. In Section 6.5 we show that by instantiating the cryptographic primitives with adequate schemes the resulting protocol can be made as efficient as state-of-the-art electronic voting systems.

6.4.2 Detailed protocol

We begin by characterizing the set of allowed votes $\mathbb{V}$. Given a set of candidates $\mathcal{V}$, we define the set of votes as $\mathbb{V} = \{v \mid v \subset \mathcal{V} \wedge |v| = n\}$ for some fixed value of n. Here, we are assuming that a voter must vote for n candidates. We can handle blank votes and undervotes by designating n different blank candidates.

Our voting protocol is built on an encryption scheme, a signature scheme and a NIZK proof system. We will require that these three schemes have a common setup algorithm Setup. The voting protocol is also built on a PRP family.

Therefore, the building blocks of our voting protocol are an encryption scheme $(\mathsf{Setup}, \mathsf{KeyGen}_{\mathsf{enc}}$, $\mathsf{Enc}, \mathsf{Dec})$, a signature scheme $(\mathsf{Setup}, \mathsf{KeyGen}_{\mathsf{sign}}, \mathsf{Sign}, \mathsf{VerifySign})$, a PRP family $F_{(\cdot)} : \mathfrak{M}_{gk} \to \mathfrak{M}_{gk}$, where $\mathfrak{M}_{gk}$ is the message space of the encryption and signature schemes defined by the output of Setup, and a NIZK proof system $(\mathsf{Setup}, \mathsf{GenCRS}, \mathsf{Prove}, \mathsf{VerifyProof})$ for the relation R defined as:

$$\begin{aligned} R = \{(gk, x, w) |\; & x = (c_1, \ldots, c_n, p_1, \ldots, p_n, pk_e, pk_s) \wedge \\ & w = (\nu_1, \ldots, \nu_n, r_1, \ldots, r_n, \sigma_1, \ldots, \sigma_n, k) \wedge \\ & (c_1, \ldots, c_n) = (\mathsf{Enc}(gk, pk_e, \nu_1, r_1), \ldots, \mathsf{Enc}(gk, pk_e, \nu_n, r_n)) \wedge \\ & (\mathsf{VerifySign}(gk, pk_s, \nu_1, \sigma_1), \ldots, \mathsf{VerifySign}(gk, pk_s, \nu_n, \sigma_n)) = (1, \ldots, 1) \wedge \\ & (p_1, \ldots, p_n) = (F_k(\nu_1), \ldots, F_k(\nu_n))\} \end{aligned}$$

The algorithms are then defined as follows:

$\mathsf{ElectionSetup}(1^\lambda)$ **:** starts by running the Setup algorithm of the cryptographic building blocks to generate the common setup gk. Then the algorithm runs $\mathsf{GenCRS}(gk)$ to generate the common reference string crs, $\mathsf{KeyGen}_{\mathsf{enc}}(gk)$ to generate a pair of encryption keys (pk_e, sk_e) and $\mathsf{KeyGen}_{\mathsf{sign}}(gk)$ to generate a pair of public/private signing keys (pk_s, sk_s). This implicitly defines the message space $\mathfrak{M}_{gk}$ for both the encryption and the signature scheme. We require that there exists an injective mapping η such that $\eta(\mathcal{V}) \subset \mathfrak{M}_{gk}$. For the sake of simplicity, we will assume that $\mathcal{V} \subset \mathfrak{M}_{gk}$. Then, for each $\nu \in \mathcal{V}$, the algorithm produces a signature on it, $\sigma_\nu = \mathsf{Sign}(gk, sk_s, \nu)$. The election public key is defined as $pk = (gk, \mathsf{crs}, pk_e, pk_s, \{\sigma_\nu\}_{\nu \in \mathcal{V}})$ and the election secret key is defined as $sk = sk_e$.

$\mathsf{Vote}(pk, v)$**:** parses pk as $(gk, \mathsf{crs}, pk_e, pk_s, \{\sigma_\nu\}_{\nu \in \mathcal{V}})$ and v as $(\nu_1, \ldots, \nu_n)$. It then samples fresh randomness $(r_1, \ldots, r_n)$ and runs $(\mathsf{Enc}(gk, pk_e, \nu_1, r_1), \ldots, \mathsf{Enc}(gk, pk_e, \nu_n, r_n))$ obtaining ciphertexts $C = (c_1, \ldots, c_n)$. Next, it selects a fresh random PRP key $k \in K_{PRP}$ and computes $(p_1, \ldots, p_n) = (F_k(\nu_1), \ldots, F_k(\nu_n))$. Finally, it computes a NIZK proof π for the relation R defined above with statement $x = (c_1, \ldots, c_n, p_1, \ldots, p_n, pk_e, pk_s)$ and witness $w = (\nu_1, \ldots, \nu_n, r_1, \ldots, r_n, \sigma_{\nu_1}, \ldots, \sigma_{\nu_n}, k)$. The output is then $b = (C, \pi, \{p_i\}_{i=1}^n)$.

$\mathsf{ValidateBallot}(BB, b)$**:** recovers pk from the bulletin board BB and parses pk as $pk = (gk, \mathsf{crs}, pk_e, pk_s, \{\sigma_\nu\}_{\nu \in \mathcal{V}})$. Upon reception of a ballot b, which is parsed as $b = (C, \pi, \{p_i\}_{i=1}^n)$, it is checked if in the bulletin board there is another ballot b' such that $c'_j = c_i$ for any $i, j \in \{1, \ldots, n\}$. If any such ballot is found, the algorithm stops and returns 0. Otherwise, the algorithm checks that the values $(p_1, \ldots, p_n)$ are distinct, returning 0 if they are not. If the values are distinct, the algorithm returns the output of VerifyProof using the statement $x = (c_1, \ldots, c_n, p_1, \ldots, p_n, pk_e, pk_s)$.

$\mathsf{Tally}(sk, BB)$**:** after each individual ballot $b \in BB$ is processed with ValidateBallot, the tally algorithm *decrypts* them and the result function is computed. The *decryption procedure* is defined as follows.

1. $(\tilde{\nu_1}, \ldots, \tilde{\nu_n}) = (\mathsf{Dec}(gk, sk_e, c_1), \ldots, \mathsf{Dec}(gk, sk_e, c_n))$ is computed.
2. It is checked that $\tilde{\nu}_1, \ldots, \tilde{\nu}_n \in \mathcal{V}$.

3. It is checked that $(\tilde{\nu_1}, \ldots, \tilde{\nu_n})$ are pairwise different.

4. If any check fails, v is assigned the value $\perp$. Otherwise, v is assigned the value $(\tilde{\nu_1}, \ldots, \tilde{\nu_n})$.

Then, ρ is applied to the resulting decryptions $\{v\}$. Note that, for each v, either $v \in \mathbb{V}$ or $v = \perp$, so ρ can be applied. The output of ρ is defined as the result and the proof of correct tabulation is defined to be the empty string ϵ

As the proof of correct tabulation is the empty string ϵ, $\mathsf{VerifyTally}$ returns 1 on any input.

6.4.3 Security proofs

Finally, we give the security properties fulfilled by our scheme. Let $(\mathsf{Setup}, \mathsf{KeyGen}_{\mathsf{enc}}, \mathsf{Enc}, \mathsf{Dec})$ be a NM-CPA secure encryption scheme, let $(\mathsf{Setup}, \mathsf{GenCRS}, \mathsf{Prove}, \mathsf{VerifyProof})$ be a NIZK proof system for relation R, and let $(\mathsf{Setup}, \mathsf{KeyGen}_{\mathsf{sign}}, \mathsf{Sign}, \mathsf{VerifySign})$ be an wEUF-CMA signature scheme and let $F_{(\cdot)} : \mathfrak{M}_{gk} \to \mathfrak{M}_{gk}$ be a PRP family for the message space $\mathfrak{M}_{gk}$ defined by Setup. Let ρ be the counting function which outputs its inputs randomly permuted and let $\mathsf{Extract}$ be the *decryption procedure* of the Tally algorithm. Then, the protocol defined in Section 6.4.2 satisfies ballot privacy, strong correctness, strong consistency and vote validatability for any candidate set $\mathbb{V}$, with respect to ρ, $\mathsf{Extract}$.

Theorem 6.4.1. *Let* $(\mathsf{Setup}, \mathsf{KeyGen}_{\mathit{enc}}, \mathsf{Enc}, \mathsf{Dec})$ *be a NM-CPA secure encryption scheme, let* $F_{(\cdot)}$ *be a PRP family and let* $(\mathsf{Setup}, \mathsf{GenCRS}, \mathsf{Prove}, \mathsf{VerifyProof})$ *be a NIZK proof system. Then, the protocol defined in Section 6.4.2 has ballot privacy.*

Proof. Recall that privacy is defined as the indistinguishability of two experiments which depend on a bit β. We will refer to them as Exp_β for $\beta \in \{0, 1\}$.

Define $\mathsf{Exp}_{\beta,0} = \mathsf{Exp}_\beta$. Consider experiments $\mathsf{Exp}_{\beta,1}$ to be the experiment which are the same as $\mathsf{Exp}_{\beta,0}$ but the challenger runs $\mathsf{SimGenCRS}$ instead of GenCRS and it runs $\mathsf{SimProve}$ instead of Prove. Finally, let $\mathsf{Exp}_{\beta,2}$ be the experiments which are identical to $\mathsf{Exp}_{\beta,1}$ but in which the challenger uses a truly random function instead of a PRP in order to cast ballots.

Due to the zero-knowledge property of the NIZK proof system, $\mathsf{Exp}_{\beta,0}$ and $\mathsf{Exp}_{\beta,1}$ are indistinguishable for $\beta \in \{0, 1\}$. Besides, $\mathsf{Exp}_{\beta,1}$ and $\mathsf{Exp}_{\beta,2}$ are indistinguishable for $\beta \in \{0, 1\}$ due to the pseudorandomness of the PRP. Now the only thing left is to prove that $\mathsf{Exp}_{0,2}$ and $\mathsf{Exp}_{1,2}$ are indistinguishable.

Consider the $\mathsf{Enc2Vote}$ scheme defined in Section 5.4, for which the following holds:

Theorem 6.4.2. *Let* $(\mathsf{Setup}, \mathsf{KeyGen}_{\mathit{enc}}, \mathsf{Enc}, \mathsf{Dec})$ *be an NM-CPA encryption scheme. Then,* $\mathsf{Enc2Vote}$ *has ballot privacy.*

Finally, we reduce the privacy of our scheme to the privacy of $\mathsf{Enc2Vote}$.

Lemma 6.4.3. *Let $\mathcal{A}^1$ be a p.p.t. adversary that interacts which challenger $\mathcal{C}$ and outputs a bit $\alpha^{\mathcal{A}_1}$ such that $|\Pr[\alpha^{\mathcal{A}_1} = 1|\mathsf{Exp}_{0,2}] - \Pr[\alpha^{\mathcal{A}_1} = 1|\mathsf{Exp}_{1,2}]|$ is non-negligible. Then, there exists an adversary $\mathcal{A}^2$ that breaks the ballot privacy property of the* Enc2Vote *scheme.*

In our reduction, $\mathcal{A}^1$ will interact with $\mathcal{A}^2$, which will act as the challenger for $\mathcal{A}^1$. At the same time, $\mathcal{A}^2$ will interact with the privacy challenger $\mathcal{C}$. The reduction is as follows:

In the **Setup** phase, $\mathcal{C}$ will run Setup, outputting gk and posting it to the bulletin board it shares with $\mathcal{A}_2$. It will also run $\mathsf{KeyGen}_{\mathsf{enc}}$, keeping the private key for itself and publishing the public key pk_e to the same bulletin board. Then, $\mathcal{A}^2$ will run the GenCRS and the $\mathsf{KeyGen}_{\mathsf{sign}}$ algorithms and will produce signatures on each voting option, posting pk_s, crs and the created signatures to the bulletin board that it has in the game with $\mathcal{A}_1$.

In the **Voting** phase, when $\mathcal{A}^1$ submits a **Vote** query, $\mathcal{A}^2$ will submit n **Vote** queries to $\mathcal{C}$, one for each pair of candidates. The challenger $\mathcal{C}$ will answer with n pairs of ciphertexts $(c_{0,1}, \ldots, c_{0,n})$ and $(c_{1,1}, \ldots, c_{1,n})$. $\mathcal{A}^2$ will then sample two pairs of random values $(p_{0,1}, \ldots, p_{0,n})$ and $(p_{1,1}, \ldots, p_{1,n})$ of the target space of the PRP. Finally, it will create ballots $b_0 = (c_{0,1}, \ldots, c_{0,n}, p_{0,1}, \ldots, p_{0,n}, \pi_0)$ and $b_1 = (c_{1,1}, \ldots, c_{1,n}, p_{1,1}, \ldots, p_{1,n}, \pi_1)$ where π_0 and π_1 will be simulated. $\mathcal{A}^2$ will post these ballots to the respective bulletin boards. Finally, when $\mathcal{A}^1$ submits a **Ballot**(b) query, $\mathcal{A}^2$ will run the ValidateBallot algorithm and will create a **Ballot**(b') for $\mathcal{C}$ with $b' = (c_1, \ldots, c_n)$ from b.

It is straightforward to see that the output of $\mathcal{A}^2$ in its interaction with $\mathcal{A}^1$ is correctly distributed, which implies that the reduction is sound.

□

In addition to ballot privacy, our protocol also satisfies strong correctness and strong consistency.

Theorem 6.4.4. *Let ρ be the counting function which outputs its inputs randomly permuted and filtering invalid votes. Let* Extract *be the algorithm defined as* $\mathsf{Extract}(b, sk) = \mathsf{Dec}(gk, sk_e, c)$. *Then, the protocol defined in Section 6.4.2 has strong consistency with respect to* Extract.

Theorem 6.4.5. *The protocol defined in Section 6.4.2 has strong correctness.*

Both theorems can be reduced to proving that the Enc2Vote satisfies those properties. The reductions are very similar to the one used to prove Lemma 6.4.3. On the other hand, Enc2Vote does satisfy strong correctness and strong consistency with respect to Extract as explained in Section 5.4. Therefore, our voting protocol also satisfies those properties.

Theorem 6.4.6. *Let ρ be the counting function which outputs its inputs randomly permuted. Let* (Setup, GenCRS, Prove, VerifyProof) *be a NIZK-PoK proof system and let* (KeyGenSign, Sign, VerifySign) *be an wEUF-CMA signature scheme. Let* Extract *be the* decryption procedure *of the* Tally *algorithm of the protocol defined in Section 6.4.2. Then, the protocol defined in Section 6.4.2 has vote validatability for any $\mathbb{V}$, with respect to ρ,* Extract.

Proof. We have already stated that our protocol satisfies strong consistency.

Let Exp_0 be the vote validatability experiment and let Exp_1 be identical to Exp_0 but instead of using GenCRS the challenger uses $\mathsf{ExtGenCRS}$. These two experiments are indistinguishable by the properties of the NIZK-PoK. Now assume that an adversary $\mathcal{A}^1$ is able to output a ballot b in the experiment Exp_1 such that $\mathsf{ValidateBallot} = 1$ and $\mathsf{Extract}(sk, b) = \perp$. Then, we build an adversary $\mathcal{A}^2$ which breaks the EUF-CMA of the signature scheme.

The reduction is straightforward: $\mathcal{A}^2$, interacting with an wEUF-CMA challenger asks for signatures on $\{\nu\}_{\nu \in \mathcal{V}}$. Then, it interacts with $\mathcal{A}^1$, posing as a vote validatability challenger. It runs all the algorithms as in the protocol but uses $\mathsf{ExtGenCRS}$, keeping the trapdoor key tk for itself, and using the answers from the wEUF-CMA challenger as the signatures on the voting options. When $\mathcal{A}^1$ outputs a ballot b, $\mathcal{A}^2$ uses $\mathsf{Extract}$ on π to obtain a witness $w = (\tilde{\nu}_1, \ldots, \tilde{\nu}_n, r_1, \ldots, r_n, \sigma_{\tilde{\nu}_1}, \ldots, \sigma_{\tilde{\nu}_n}, k))$ such that $(gk, x, w) \in R$. This means that $\mathsf{VerifySign}(gk, pk_s, \tilde{\nu}_i, \sigma_{\tilde{\nu}_i}) = 1$ for $i \in \{1, \ldots, n\}$. $\mathsf{Extract}(gk, sk, b)$ might return $\perp$ either because (i) some $\mathsf{Dec}(gk, sk_e, c_i) = \perp$, (ii) some $\tilde{\nu}_i = \tilde{\nu}_j$ for $i \neq j$ or (iii) some $\tilde{\nu}_i \notin \mathcal{V}$. However, (i) and (ii) are ruled out due to w being a valid witness, so the only possibility is (iii). Then, $\mathcal{A}^2$ can submit $(\tilde{\nu}_i, \sigma_{\tilde{\nu}_i})$ as its wEUF-CMA forgery. □

6.5 Concrete instantiation

We now give a concrete instantiation of the voting protocol given above. In order to give the concrete instantiation, we just need to define which encryption scheme, signature scheme, pseudo-random permutation family and NIZK-PoK scheme the protocol will use. With regard to our instantiation, the candidates will be encoded as n *randomly sampled* elements of a group $\hat{\mathbb{G}}$ and we will require that $|\mathcal{V}|$ is polynomial in the security parameter.

The Setup algorithm will consist of a group generation algorithm $\mathcal{PG}$ which outputs type-III bilinear groups $(p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h})$ as common setup gk. We will assume that the SXDH assumption introduced in Section 2.2 holds relative to $\mathcal{PG}$.

Encryption scheme

The protocol will use the Signed ElGamal (introduced in Section 2.4.1) in $\hat{\mathbb{G}}$, which is NM-CPA secure if the DDH problem is hard $\hat{\mathbb{G}}$, which is implied by the SXDH assumption.

Signature scheme

The signature scheme that we will use is the structure-preserving signature scheme given in [AGOT14] and explained in Section 2.5.1. The structure-preserving signature scheme uses the common Setup algorithm as required.

Pseudo-Random Permutation family

We will define the set of candidates, $\mathcal{V}$, as a set containing n randomly sampled group elements from $\hat{\mathbb{G}}$, where n is polynomial in the security parameter. This allows us to define the pseudo-random function $F_k : \hat{\mathbb{G}} \to \hat{\mathbb{G}}$ where $F_k(\hat{g}) = k\hat{g}$ and $k \in \mathbb{Z}_p^*$. As we assume that the Decisional Diffie-Hellman problem is hard in $\hat{\mathbb{G}}$, this function family is pseudo-random when we restrict the input to $\mathcal{V}$. [2]

Non-Interactive Zero-Knowledge Proof of Knowledge

Finally, we have to give the NIZK-PoK scheme that we will use. We will use Groth-Sahai proofs framed as a Commit-and-Prove scheme as explained in Section 4. In this case, we can use the common Setup as the setup for the Commit-and-Prove scheme.

In addition, the Commit-and-Prove scheme explained in Section 4 allows us to treat the ElGamal encryption of a value as a commitment of type "encryption", where the randomness used for the encryption is the randomness used for the commitment. The encryption scheme is thus embedded into the Commit-and-Prove scheme, instead of being an independent scheme, as assumed in the general construction. However, we can still adapt the security proof to keep it sound as we now describe.

We will consider the conjunction of two proofs. The first one is a zero-knowledge proof for the language defined by the relation

$$\begin{aligned} R_1 = \{(gk, x, w) |\ & x = (c_1, \dots, c_n, pk_e, pk_s) \wedge \\ & w = (\nu_1, \dots, \nu_n, r_1, \dots, r_n, \sigma_1, \dots, \sigma_n) \wedge \\ & (c_1, \dots, c_n) = (\mathsf{Enc}(gk, pk_e, \nu_1, r_1), \dots, \mathsf{Enc}(gk, pk_e, \nu_n, r_n)) \wedge \\ & (\mathsf{VerifySign}(gk, pk_s, \sigma_1, \nu_1), \dots, \mathsf{VerifySign}(gk, pk_s, \sigma_n, \nu_n)) = (1, \dots, 1)\} \end{aligned}$$

For this proof, the prover computes a commitment to each value of the signature and builds proofs for satisfiability of the verification equations.

We now need to see that the Commit-and-Prove scheme given in Section 4 is Zero-Knowledge for the language defined by R_1. In other words, we need to see that exists a simulator. As seen in Section 4, this reduces to check that there are no terms in pairing product equations which prevent simulation. Those terms are pairings where in each side of the pairing there is either a public, non-equivocable value[3] or a value which commitment type is "encryption". Going back to the verification equations of the signature scheme, we see that there are none of these terms. Therefore, there exists a simulator for the R_1 statement.

[2] Technically, it is a Pseudo-Random Function [GGM86] from $\mathcal{V}$ to $\hat{\mathbb{G}}$ where $F_{(\cdot)}$ is injective for any $k \in \mathbb{Z}_p^*$. Therefore, an adversary restricted to only evaluate the function in points from $\mathcal{V}$ can not distinguish those evaluations from randomly sampled elements, which is sufficient for the security reduction to work.

[3] In Section 4 we defined equivocable values as the generators of the group. However, it can be seen that values for which the simulator knows the discrete logarithm w.r.t. the generator of the group are also equivocable.

The second zero-knowledge proof is defined by the relation

$$\begin{aligned} R_2 = \{(gk, x, w) |\; & x = (c_1, \ldots, c_n, p_1, \ldots, p_n, pk_e) \wedge \\ & w = (\nu_1, \ldots, \nu_n, r_1, \ldots, r_n, k) \wedge \\ & (c_1, \ldots, c_n) = (\mathsf{Enc}(gk, pk_e, \nu_1, r_1), \ldots, \mathsf{Enc}(gk, pk_e, \nu_n, r_n)) \wedge \\ & (p_1, \ldots, p_n) = (F_k(\nu_1), \ldots, F_k(\nu_n))\} \end{aligned}$$

For the proof for the relation R_2, we will consider the multi-exponentiation equations $k\hat{\nu}_i = \hat{p}_i$, where $\hat{\nu}_i$ and k are secret values. The prover computes a commitment on k and builds a proof for satisfiability of this equation using c_i as a commitment to $\hat{\nu}_i$. Both the commitments on the signatures and the proofs will be included in π. As noted in Section 4, multi-exponentiation equations are always simulatable.

Note that the Commit-and-Prove scheme given in Section 4 is not extractable for exponents such as k. However, the proof of our scheme having vote validatability only needs to extract the values $\hat{\nu}_1, \ldots, \hat{\nu}_n$ and their corresponding signatures.

6.5.1 Efficiency

When counting the number of elements required by the Signed ElGamal encryption, the images of the PRP and the NIZK-PoK, we get that a ballot consists of $19n$ elements in $\hat{\mathbb{G}}$, $12n + 2$ elements in $\check{\mathbb{H}}$ and $2n$ elements in $\mathbb{Z}_p$. The cost is linear in n (the number of candidates encoded in each vote). Moreover, the constant factor is relatively small.

6.6 Conclusions

We have formalized the definition of vote validatability in order to give an accurate meaning to *avoid voters from casting invalid votes*, both if done in purpose or as a consequence of a software bug. Besides creating a construction based on generic building blocks and general-purpose zero-knowledge proofs, we have provided a concrete instantiation. We have shown that its efficiency fits into the computational capacity of voting devices currently used in electronic elections.

There are other alternatives which may improve the performance of our construction achieving the same security properties. First, a cryptographic accumulator could be used to prove that candidates are valid. This approach could reduce the length of the ballot. Second, much of the cost of the ballot comes from the NIZK-PoK proof for the language defined by R_1. A choice of a different structure-preserving signature scheme might improve the efficiency of our system.

Chapter 7

Universal Cast-as-Intended Verifiability

This paper is mainly based on the research published in [EGHM16], coauthored with Sandra Guasch, Javier Herranz and Paz Morillo.

7.1 Motivation

Intuitively, it seems that cast-as-intended verification can only be performed by the voter who cast the vote, given that she made her selections in secret. However, leaving the responsibility of this verification to the voter may not be effective at all.

On the one hand, we might think of systems which make the cast-as-intended verification mandatory in order to cast a vote. The problem is that current cast-as-intended verification systems present important drawbacks: verification mechanisms are not very usable and, in most cases, voters have to engage in highly interactive protocols and/or be able to perform complex computations. Therefore, mandatory cast-as-intended verification would disenfranchise less skilled voters.

On the other hand, we could allow the cast-as-intended verification to be an optional step which the voter can do before casting the vote. This does not solve the problem though, as targeted attacks against non-skilled voters, who will probably not use the verification system, can succeed undetectably. Even if all voters use the same voting device and voter authentication does not take place in the voting device, the voting device could heuristically detect the behaviour of the voter and guess whether she will use the verification mechanism.

7.2 Our contributions

In this contribution, we present a universal cast-as-intended verification protocol, i.e., a protocol which cast-as-intended verification is not restricted to the voter anymore. This allows to deploy mechanisms for auditing all the cast votes in an extensive form, and ensure no tampering attacks happen. Once the voter has cast a ballot, our protocol does not require her to take any further step to verify that her cast ballot corresponds to her chosen candidates.

This is a change of paradigm with respect to current voting protocols, as in our new protocol the verification of the content of the encrypted votes does not rely on the willingness or skills of the voters. Indeed, it can now rely on third parties (e.g., election auditors) without compromising voters' privacy.

The main idea of our protocol is the following: a voter registers to vote with a set of registrars of her choice. Each registrar generates a pair of public-secret values for each voting option in the election. The secret values are provided to the voter and the public ones are published together with their relation to the voting options. During the voting phase, the voter provides her selected voting option and a subset of the secret values she received during registration to the voting device. The voting device then encrypts the voter's selections and creates a NIZK-PoK, which will be valid only in the case the voting device encrypted what the voter selected.

Thanks to the zero-knowledge property of the proofs, they can be publicly verified while maintaining the voter's privacy. Therefore, we can say that the system provides *universal cast as intended verifiability*.

The security of our scheme relies on some assumptions, which are presented in Section 7.3.2 but for which we provide a brief overview here. The voting device, in charge of encrypting the voter's choice, is trusted for privacy (i.e., it is assumed to not to leak any information about the voter's selection) but it is not trusted for integrity, so the verifiability of the scheme does not depend on its honesty. The registrars are the entities in charge of providing some secret values to the voter for the universal cast-as-intended verification. We assume that each voter is able to select any subset of registrars on which she relies. We assume that corrupt registrars are honest but curious. In addition, when considering verifiability, we assume that at least one of the chosen registrars is completely honest.

7.2.1 Related work

At a first glance one would say that our voting protocol is similar to any Code Voting protocol [Cha01a]: in both systems voters introduce codes in their voting devices. However, there are important differences between our voting protocol and any Code Voting protocol. Unlike Code Voting systems, privacy of our voting protocol does not depend on the secrecy of the voting codes. In addition, not only our voting protocol is (universally) cast-as-intended verifiable but it can be extended to be end-to-end verifiable by using known techniques such as a Bulletin Board [GV10] and a verifiable Mix-Net [SK95]. As far as we know, the only end-to-end verifiable Code Voting protocol (with individual cast-as-intended verifiability) is VeryVote [JRF09], in which verifiability is achieved at the cost of making the scheme non-private to the voting server. Finally, note that our voting protocol does not attempt to solve the secure platform problem [GNR$^+$01] and, in particular, it can be implemented with a user-friendly point and click voting interface (even though voters still need to introduce voting codes).

7.3 Electronic voting definitions

7.3.1 Syntactical definition

We now give the syntax of a voting scheme with universal cast-as-intended verifiability. This definition is based on single-pass voting schemes, as defined in [BPW12b]. This kind of schemes are characterized by the fact that voters interact with the system only by submitting their ballots. Since we will add functionalities to the voting scheme, we need to modify and extend the definitions given in Section 5.3.

As defined in [BPW12b], a voting scheme has the following participants: the *Election Authorities* are in charge of setting up the election, computing the tally and publishing the results; the *Voters* participate in the election by choosing their preferred options; and the *Bulletin Board Manager* receives, processes and publishes the ballots received, as well as other public information.

In addition, we also consider the following participants: the *Registrars*, from which a subset is chosen by each voter to register with. The registrars are responsible for providing all the information needed to vote and, in particular, the information that will provide the UCIV property, to the voters that chose them as registrars. We also explicitly consider the *Voting Device* as a participant. As opposed to *Voters*, who only choose their preferred voting option, the *Voting Device* is in charge of casting a ballot given this voting option. This distinction is important when introducing the concept of universal cast-as-intended verifiability.

We assume that non-cryptographic election specifications such as the set of voting options $\mathbb{V}$, the set of *Registrars* RID or the set of voters are fixed in advance by the *Election Authorities*. Further we assume a counting function $\rho : (\mathbb{V} \cup \{\perp\})^* \rightarrow R$ is given, where $\mathbb{V}$ is the set of voting options, $\perp$ denotes an invalid vote and R is the set of results.

For sake of simplicity, we will assume that there is only one Election Authority. Detailed trust assumptions for Election Authorities are further discussed in Section 7.3.2.

The voting scheme is characterized by the following algorithms:

$\mathsf{ElectionSetup}(1^\lambda)$**:** is a protocol executed by the Election Authority. On input a security parameter 1^λ, it generates and outputs an election public/private key pair (pk, sk). In addition, it defines the space of secret Universal Cast-As-Intended Verification (UCIV from now on) information SVI, the space of voting option-dependent secret UCIV information $\widetilde{SVI}$, the space of public UCIV information PVI and a family of functions $\sigma_\cdot : SVI \rightarrow \widetilde{SVI}$ such that, for each voting option $v \in \mathbb{V}$ the function σ_v maps the secret UCIV information to some voting option-dependant secret UCIV information (*The function σ_v will need to be evaluated by the voter, so it should be a relatively simple one. In our scheme the function will consist on selecting some elements from a given set*).

$\mathsf{Register}(vid, rid, \mathbb{V}, pk)$**:** is run by registrar rid. It takes as input a voter identity vid, the set of voting options $\mathbb{V}$ and the election public key pk. It outputs partial secret UCIV information $s_{uciv}^{vid,rid} \in SVI$ and partial public UCIV information $p_{uciv}^{vid,rid} \in PVI$.

Let us define the secret UCIV information as $s_{uciv}^{vid} = \{s_{uciv}^{vid,rid}\}_{rid \in RID[vid]}$ *and the public UCIV information as* $p_{uciv}^{vid} = \{p_{uciv}^{vid,rid}\}_{rid \in RID[vid]}$*, where* $RID[vid]$ *is the set of registrars with whom the voter* vid *registered.*

$\mathsf{Vote}(vid, \sigma_v(s_{uciv}^{vid}), p_{uciv}^{vid}, pk, v)$**:** is a probabilistic protocol run by voting devices. It receives as input the voter identity vid, a voting option $v \in \mathbb{V}$, the function σ_v evaluated component-wise on the secret UCIV information $s_{uciv}^{vid} \in SVI^{|RID[vid]|}$, the public UCIV information $p_{uciv}^{vid} \in PVI^{|RID[vid]|}$, the election public key pk and outputs a ballot b.

$\mathsf{ValidateBallot}(BB, b, vid)$**:** is run by the bulletin board manager. It receives as input a bulletin board BB, a ballot b and a voter identity vid and outputs either success (1) or reject (0).

$\mathsf{Tally}(BB, sk)$**:** is run by the Election Authority. It takes as input a bulletin board BB and the election secret key sk and outputs a result $r \in R$ and a correct tabulation proof Π.

$\mathsf{VerifyTally}(BB, r, \Pi)$**:** on input the bulletin board BB, the tally r and a proof of correct tabulation Π it outputs either accept (1) or reject (0).

Finally, the scheme is executed as follows:

1. In the *configuration* phase, the Election Authority runs the ElectionSetup algorithm. The Election Authority publishes the election public key pk on the bulletin board BB and keeps the election secret key sk.

2. In the *registration* phase, each voter can register with as many registrars as she wants to. Each registrar selected by the voter then runs the Register algorithm, provides the partial secret UCIV information $s_{uciv}^{vid,rid}$ to the voter, and publishes the partial public UCIV information $p_{uciv}^{vid,rid}$ to the bulletin board.

3. In the *voting* phase the voter vid chooses a voting option v, evaluates $\sigma_v(s_{uciv}^{vid})$ (where s_{uciv}^{vid} is defined as the set of all partial secret UCIV information) and provides $(vid, v, \sigma_v(s_{uciv}^{vid}))$ to the voting device. The voting device takes the election public key pk and the public UCIV information p_{uciv}^{vid} from the bulletin board and runs the Vote algorithm, producing a ballot b. The ballot b and the voter's identity vid are then sent to the bulletin board. Upon reception of the ballot, the bulletin board manager executes the ValidateBallot algorithm. In case the output is 1, the ballot is published in the bulletin board, otherwise the ballot is discarded and the voter is notified accordingly.

4. In the *counting* phase the Election Authority runs the Tally algorithm using the election private key sk and the information in the bulletin board, including the ballots. The output of the Tally algorithm is published on the bulletin board.

A voting system as defined above is correct if, when the four phases are run with all the participants behaving correctly, the result r output by the Tally algorithm is equal to the evaluation of the counting function ρ on the voting options corresponding to the ballots cast by the voters.

7.3.2 Security definitions

In this section we give the security definitions for an electronic voting scheme. In particular, we define ballot privacy and universal cast-as-intended verifiability.

Assumptions on voting devices, election authorities and registrars

The security of our electronic voting scheme will rely on some assumptions. Even though they will be properly formalized in this section, we first give an informal intuition.

When considering voting devices, we distinguish between privacy and integrity. In order to guarantee privacy, we assume that the voting device behaves properly by correctly encrypting the voter's choice and not leaking any information. This is a common assumption in electronic voting schemes where the voter can use a point and click interface to make her choices, such as Helios [Adi08, BCP^{+}11]. On the other hand, we do not trust the voting device at all when it comes to integrity, since we assume it could try to change the voter's choice prior to encryption. In the same way, the verifiability of the scheme does not rely on the voting device being honest.

We will assume that there is only one election authority, which is trusted for privacy. Secret sharing and multi-party computation techniques can be easily deployed, as done in [CGS97], to overcome this limitation so that privacy is guaranteed as long as a subset of the election authorities are trusted. The electoral authorities do not need to be trusted for our new verifiability property. Indeed, in the UCIV security definition we assume that the election private key is leaked to the adversary.

In addition, we will also assume that any common reference strings are generated by a single, trusted authority. This is a strong assumption, which can be solved by using the multi-string model defined by Groth and Ostrovsky in [GO14]. In this model, several parties provide their common reference strings and security relies on assuming that a threshold of such parties are honest. We consider that directly using the multi-string model would make the description of the protocol utterly complex and, on the other hand, it would not add much value to this contribution.

When considering registration authorities, we will consider that they are producing the UCIV information correctly. In particular, under these assumptions there should be no claims from voters about having received corrupted UCIV information from a malicious registrar. We allow each voter to choose which registrars she registers with: she might register with only one registrar or with all of them. In terms of privacy, we will assume that all the secret information generated by the registrars is leaked to the adversary. When defining UCIV, we will assume that at least one of the registrars chosen by the voter is completely honest. The other registrars will be assumed to be honest-but-curious: they will not cheat, but they might leak information, including any covert channel they might use.

We will finally assume that registrars only publish on the bulletin board UCIV information corresponding to a registration from a voter. This can be ensured by also posting some sort of registration request produced by the voter.

Ballot privacy

Intuitively, a voting system has ballot privacy if an adversary with access to the bulletin board is not able to guess what voting options the voters chose. We adopt the formalization given in [BCG$^+$15], where they give a definition correcting the flaws of previous definitions.

Ballot privacy is defined using two experiments between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The goal of the adversary is to distinguish between the two experiments. In both experiments, we let the adversary corrupt voters and submit ballots on their behalf. In addition, for each honest voter the adversary can also specify two votes to be used for casting her ballot. The votes which will be used to cast the honest voters' ballots will depend on which experiment is taking place. The goal of the adversary is to distinguish between both experiments i.e., to distinguish which votes were used to cast the honest voters' ballots. As revealing the "true" tally would easily allow the adversary to distinguish between the experiments, the same tally is always shown to the adversary, regardless of which vote was used to cast honest voters' ballots.

For compactness, we present the two experiments as a single experiment depending on a bit β. The experiments are parametrized by the set of voting options $\mathbb{V}$ and an algorithm $\mathsf{SimProof}(BB, r)$ such that, given a bulletin board and a result simulates a proof of correct tabulation (in this case, the $\mathsf{ElectionSetup}$ algorithm will need to be modified so that proofs can be simulated).

1. **Setup phase**. $\mathcal{C}$ sets up two empty bulletin boards BB_L and BB_R. It runs the $\mathsf{ElectionSetup}(1^\lambda)$ protocol to obtain the election public key pk and the election private key sk. It then posts pk on both bulletin boards. The adversary is given read access to either BB_L if $\beta = 0$ or BB_R if $\beta = 1$. In addition, $\mathcal{C}$ initializes an empty list ID.

2. **Registration phase**. The adversary may make one type of query.

   - **Register**(vid, rid) query. The adversary provides a voter identity and a registrar identity such that $(vid, rid) \notin ID$. The challenger runs $\mathsf{Register}$ on inputs $(vid, rid, \mathbb{V}, pk)$ to generate the partial public UCIV information $p_{uciv}^{vid,rid}$ and the partial secret UCIV information $s_{uciv}^{vid,rid}$. $\mathcal{C}$ provides both $p_{uciv}^{vid,rid}$ and $s_{uciv}^{vid,rid}$ to $\mathcal{A}$ and $p_{uciv}^{vid,rid}$ is published on both bulletin boards. The pair (vid, rid) is added to ID.

3. **Voting phase**. The adversary may make two types of queries.

- **Vote**(vid, v_L, v_R) queries. The adversary provides a voter identity vid such that there is at least one rid for which $(vid, rid) \in ID$ and two votes $v_L, v_R \in \mathbb{V}$. The challenger runs $\mathsf{Vote}(vid, \sigma_{v_L}(s^{vid}_{uciv}, v_L), p^{vid}_{uciv}, pk)$, which outputs b_L and $\mathsf{Vote}(vid, \sigma_{v_R}(s^{vid}_{uciv}), p^{vid}_{uciv}, pk, v_R)$ which outputs b_R (in this case, $s^{vid}_{uciv}, p^{vid}_{uciv}$ are the set of partial public/secret UCIV information corresponding to the registrars rid such that $(vid, rid) \in ID$). $\mathcal{C}$ then obtains new versions of the boards BB_L and BB_R by running $\mathsf{ValidateBallot}(BB_L, b_L, vid)$ and $\mathsf{ValidateBallot}(BB_R, b_R, vid)$ and updating the boards accordingly.
- **Ballot**(b, vid) queries. These are queries made on behalf of corrupt voters. Here the adversary provides a ballot b and an identity vid such that there is some rid with $(vid, rid) \in ID$. The challenger runs $\mathsf{ValidateBallot}(BB_L, b, vid)$ and if the process accepts it also runs $\mathsf{ValidateBallot}(BB_R, b, vid)$ and updates the boards accordingly.

4. **Tallying phase**. The challenger evaluates $\mathsf{Tally}(BB_L, sk)$ obtaining the result r and the proof of correct tabulation Π. If $\beta = 0$, the challenger posts (r, Π) on the bulletin board BB_L. If $\beta = 1$, the challenger runs $\mathsf{SimProof}(BB_R, r)$ obtaining a simulated proof Π' and posts (r, Π') on the bulletin board BB_R.
5. **Output**. The adversary $\mathcal{A}$ outputs a bit α.

We say that a voting protocol for $(\mathbb{V}, R, \rho)$ as defined in Section 7.3.1 provides ballot privacy if there exists an algorithm $\mathsf{SimProof}$ such that for any p.p.t. adversary $\mathcal{A}$, the following advantage is negligible in the security parameter λ.

$$\mathbf{Adv}^{\mathsf{priv}}_{\mathsf{SimProof}}(\lambda) := |\Pr[\alpha = 1|\beta = 1] - \Pr[\alpha = 1|\beta = 0]|$$

We want to remark that in case of honest voters the ballots are properly encrypted. In other words, this implies that the voting devices used to cast those ballots use *good* randomness and do not leak information about the randomness used. In addition, the ballot privacy does *not* rely on the adversary not having access to the secret UCIV information.

In addition to voting privacy we will also require that voting protocols satisfy strong consistency and strong correctness as defined in Section 5.3.4.

Universal cast-as-intended verifiability

Intuitively, a voting system satisfies the cast-as-intended property if a corrupt voting device is not able to cast a ballot for a voting option different to the one chosen by the voter. This should hold as long as the voter is honest. If the voter is malicious no guarantees can be given besides the fact that the ballot must correspond to at most one voting option (i.e., it could also correspond to an invalid voting option, which will not be counted). We define cast-as-intended on a per-ballot basis, not considering the tallying phase inside the definition.

Universal cast-as-intended verifiability is defined as an experiment between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. In this experiment, the adversary may corrupt registrars, voters or voting devices. The goal of the adversary is to cast ballots on behalf of a non-corrupt voter so that the ballot does not extract to the voting option chosen by the voter. Here, the *extract* algorithm must be a meaningful one, we require it to be one with which the voting scheme is strongly consistent. The experiment is parametrized by the set of voting options $\mathbb{V}$ and an algorithm $\mathsf{Extract}(b, sk)$ such that, given a ballot and the election private key returns a vote or $\perp$ denoting an invalid vote.

1. **Setup phase**. The challenger sets up an empty bulletin board BB and runs the $\mathsf{ElectionSetup}(1^\lambda)$ protocol to obtain the election public key pk and the election private key sk, posts pk on the board and gives (pk, sk) to the adversary. The adversary is given read access to BB. The adversary $\mathcal{A}$ announces the (strict) subset of registrars $\widetilde{RID} \subset RID$ which he corrupts so that they become honest-but-curious. In addition, $\mathcal{C}$ initializes three empty lists ID_R, ID_P, ID_F. For convenience, we define $ID = ID_P \cup ID_F$

2. **Registration phase**. The adversary may make one type of query.

   - **Register**(vid, rid) query. The adversary provides a voter identity and a registrar identity such that $(vid, rid) \notin ID_R$. The challenger runs $\mathsf{Register}(vid, rid, \mathbb{V}, pk)$ to generate the partial public UCIV information $p_{uciv}^{vid,rid}$ and the partial secret UCIV information $s_{uciv}^{vid,rid}$. If $rid \in \widetilde{RID}$, $\mathcal{C}$ provides both $p_{uciv}^{vid,rid}$ and $s_{uciv}^{vid,rid}$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ only provides $p_{uciv}^{vid,rid}$ to $\mathcal{A}$. In both cases, $p_{uciv}^{vid,rid}$ is published on the bulletin board and (vid, rid) is added to ID_R.

3. **Voting phase**. The adversary may make two types of queries.

   - **CorruptVotingDevice**(vid, v_{vid}) queries. The adversary provides a voter identity $vid \notin ID$ such that $(vid, rid) \in ID_R$ for some $rid \notin \widetilde{RID}$ and a voting option v_{vid} corresponding to such identity. Then, $\mathcal{C}$ provides $\sigma_{v_{vid}}(s_{uciv}^{vid,rid})$ for all $(vid, rid) \in ID$ to $\mathcal{A}$. The challenger adds vid to ID_P.
   - **CorruptVoter**(vid) queries. The adversary provides a voter identity $vid \notin ID$ such that $(vid, rid) \in ID_R$ for some rid. Then, $\mathcal{C}$ provides $s_{uciv}^{vid,rid}$ for all $(vid, rid) \in ID$ to $\mathcal{A}$. The challenger adds vid to ID_F.

4. **Output**. The adversary submits a pair (b^*, vid^*). The output of the experiment is a bit δ, which is defined as 1 if (i) $vid^* \in ID_P$, (ii) $\mathsf{ValidateBallot}(BB,\ b^*, vid^*) = 1$ and (iii) $\mathsf{Extract}(b^*, sk) \neq v_{vid^*}$, where v_{vid^*} is the voting option submitted by the adversary in the PartialCorruptVoter query for vid^*. $\delta^{\mathbb{V}}$, is defined as 0 in any other case.

We say that a voting protocol for $(\mathbb{V}, R, \rho)$ as defined in Section 7.3.1 has universal cast-as-intended verifiability if there exists an algorithm $\mathsf{Extract}$ such that the following two conditions hold:

(i) the voting protocol is strongly consistent with respect to $\mathsf{Extract}$, as defined in Section 5.3.4.

(ii) for any p.p.t. adversary $\mathcal{A}$, the following advantage is negligible as a function of λ.

$$\mathbf{Adv}^{\mathsf{uciv}}_{\mathsf{Extract}}(\lambda) := \Pr[\delta = 1]$$

We want to remark that the universal cast-as-intended verifiability property does not rely on the secrecy of the private election key.

7.4 Core voting protocol

7.4.1 Overview

In this section, we present our new voting protocol. We will not take into account usability issues from a voter's point of view; we will take care of these issues in the voting systems proposed in Section 7.7. We think that by splitting the core protocol from the voting systems built on top of it the reader can get a clearer picture of our solution.

Our protocol is a mix-net based voting protocol. In mix-net based protocols, the voters encrypt their votes (we will assume that each selection is encrypted individually), then all the ciphertexts submitted by the voters are shuffled and re-encrypted in several nodes and then the shuffled ciphertexts are decrypted. We will work with a simplified scheme where votes are only shuffled and decrypted once.

The goal of our work is to provide a protocol which provides ballot privacy and universal cast-as-intended verifiability. Other requirements of voting schemes such as the assurance of the authorship of the vote (an eligible voter), recorded-as-cast verifiability and counted-as-recorded verifiability are not considered in this work, since other measures can be built around the protocol in order to fulfill them. For instance, digital signatures, public bulletin boards or verifiable mix-nets could be used to satisfy the mentioned properties respectively.

The main idea of the core protocol is the following: during the registration phase, a voter registers to vote with a set of registrars of her choice. Each one of the selected registrars generates a secret value for each candidate of the election, and provides them to the voter. A one-way function will be applied to each secret value and the resulting images will be made public (maintaining the relation with the candidates).

Once the voter has selected her voting option by using a voting device, she will provide a subset of the secret values (previously unknown) to the voting device. Then the voting device will encrypt the voting option, and will create a NIZK-PoK. By carefully choosing which secret values the voter discloses, the voting device will only be able to create a valid NIZK-PoK if it really encrypted the voter's selections.

The NIZK-PoK used in the protocol can be publicly verified, since the voter's privacy is maintained in such case due to the zero-knowledge property of the proofs. Therefore, the system provides *universal cast* *as intended verifiability* since the proofs can be universally verified and this validation is successful only in the case the voting device encrypted the voter's selections.

A sketch on how this NIZK-PoK works is the following: for sake of simplicity, assume that the voter can only select one candidate and there is only one registrar. In the voting phase, the voter will provide the voting device the secret values whose images are associated to the candidates she did not choose. The voting device will then create, for each possible candidate, a NIZK-PoK of the statement "either this candidate is encrypted in the ciphertext *or* I know the pre-image of the public value which corresponds to it". Without further information, the voting device will be able to create all the proofs as long as it encrypted the voter's selected candidate. When dealing with several registrars, the secret and public values will be defined as the homomorphic aggregation of the secret and public values provided by each registrar.

The proofs also guarantee that, as long as the voter behaved honestly, the encrypted vote contains a valid candidate. This might be useful to discern whether a malformed vote was intentionally created by a malicious voter or by a malicious voting device (used by an honest voter). Note that, as we are giving a mix-net based protocol, if both the voter and the voting device are malicious and submit a vote containing a non-valid candidate, this vote will be anyway discarded once the votes have been mixed and decrypted, prior to being added to the count. Therefore, this is not an attack against the system.

7.4.2 2-cnf-proof of knowledge

In this section, we define the language for which the NIZK-PoK will be created. Above we said that one NIZK-PoK would be given for each voting answer, what we will actually do is to give a single proof for all the voting answers, which will achieve the same behaviour. We will now describe the relation for which we will construct the final NIZK-PoK.

In Section 2.4 we defined an encryption scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with an associated message space $\mathfrak{M}_{gk}$, ciphertext space $\mathfrak{C}_{gk}$ and randomness space $\mathfrak{R}_{gk}$. Consider the relation

$$R_{enc} = \{(gk, (c, m), r) | m \in \mathfrak{M}_{gk}, c \in \mathfrak{C}_{gk}, r \in \mathfrak{R}_{gk}, c = \mathsf{Enc}(gk, pk, m; r)\},$$

which consists of the tuples of setups, ciphertexts, messages and randomness such that the ciphertext is an encryption of the message using the specified randomness. Note that, in this relation, the statement is the pair ciphertext - message, while the witness is the randomness.

Also consider that we have a homomorphic one-way function F with an associated input space X and output space Y, both of them additive groups. Consider the relation $R_{ow} = \{(y, x) | x \in X, y \in Y, y = F(x)\}$ which consists of the pairs of values such that the first value is the image of the second one under the one-way function F. In this case, the image value (i.e., the first element, y) is the statement while the pre-image (i.e., the second element, x) is the witness.

The relation for which we will make the NIZK-PoK is the following one:

$$\begin{aligned} R_{\text{2-}cnf} = \{ ((gk, (c, (m_1, \dots, m_n), (y_1, ..., y_n)), (r, (x_1, \dots, x_n))) | \\ (((gk, c, m_1); r) \in R_{enc} \vee (x_1, y_1) \in R_{ow}) \wedge \\ (((gk, c, m_2); r) \in R_{enc} \vee (x_2, y_2) \in R_{ow}) \wedge \\ \cdots \qquad\qquad\qquad \wedge \\ (((gk, c, m_n); r) \in R_{enc} \vee (x_n, y_n) \in R_{ow}) \} \end{aligned}$$

This way of presenting the relation $R_{\text{2-}cnf}$ allows us to analyze it easily: the relation takes a statement, which is a ciphertext, a set of messages and a set of image values, and a witness, which is some randomness and a set of pre-images. The tuple of setup, statement and witness belongs to the language iff every predicate in the *AND* clause is satisfied. This is, we require that for every $i \in \{1, \dots, n\}$ the predicate $((gk, (c, m_i), r) \in R_{enc} \vee (x_i, y_i) \in R_{ow})$ is satisfied. This predicate perfectly captures the intuition given above: either the ciphertext encrypts the message m_i or a pre-image of y_i is known. The description of the language $R_{\text{2-}cnf}$ will also allow us to generate a NIZK-PoK from the composition of simpler NIZK-PoKs for the relations R_{enc} and R_{ow}.

As explained in Section 2.7, NIZK-PoKs can be constructed for any relation which can be verified in polynomial-time in the length of the statement. The relation $R_{\text{2-}cnf}$ described above is of such type, so we directly have a NIZK-PoK system for such relation. In Section 7.6.1 we present efficient NIZK-PoK systems for the $R_{\text{2-}cnf}$ relation.

7.4.3 Detailed protocol

Our protocol implements a voting scheme where the counting function ρ is the multiset function as defined in [BCG$^+$15]. This function returns a random permutation of its input, filtering invalid votes. This is the case for all mix-net based protocols.

We will start by describing the protocol for single-mark ballots. In Section 7.4.4 we present an extension for multiple-mark ballots. For the sake of simplicity, we will consider blank candidates as additional voting options the voter can choose, both for single-mark and for multiple-mark systems.

Our protocol is built on an encryption scheme, a homomorphic one-way function and a NIZK-PoK proof system. We will require that the schemes have the same setup algorithm.

Therefore, the building blocks of our protocol are an encryption scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, a homomorphic one-way function F and a NIZK-PoK $(\mathsf{Setup}, \mathsf{GenCRS}, \mathsf{Prove}, \mathsf{VerifyProof})$ for the relation $R_{\text{2-}cnf}$ defined in Section 7.4.2.

We define the partial secret UCIV information for a voter as a set of pre-images $(x_1^{vid,rid}, \dots, x_n^{vid,rid})$ of F, one for each element in $\mathbb{V}$ (the set of voting options), and the partial public UCIV information as $(y_1^{vid,rid}, \dots, y_n^{vid,rid}) = (F(x_1^{vid,rid}), \dots, F(x_n^{vid,rid}))$, the images of each element of the secret UCIV information. Each pair $(x_i^{vid,rid}, y_i^{vid,rid})$ is related to the corresponding voting option v_i, and the relation between $y_i^{vid,rid}$ and v_i is public.

By defining the secret UCIV information as $(x_1^{vid}, \dots, x_n^{vid}) = (\sum_{rid} x_1^{vid,rid}, \dots, \sum_{rid} x_1^{vid,rid})$ and $(y_1^{vid}, \dots, y_n^{vid}) = (\sum_{rid} y_1^{vid,rid}, \dots, \sum_{rid} y_1^{vid,rid})$ it will still be satisfied that $y_i^{vid} = F(x_i^{vid})$.

The function σ_{v_i} has as input the set of all pre-images $(x_1^{vid,rid}, \dots, x_n^{vid,rid})$ and outputs the same values except for $x_i^{vid,rid}$. Note that this implicitly defines $SVI, \widetilde{SVI}, PVI$. As all this information is only determined by F, there is no need to consider the specification of $SVI, \widetilde{SVI}, PVI, \sigma$ as part of the ElectionSetup algorithm.

The protocol consists of the following algorithms:

ElectionSetup(1^λ)**:** the algorithm first runs Setup to generate a setup gk. Then it runs GenCRS(gk) to generate the common reference string crs and KeyGen(gk) to generate a pair of public/private encryption keys (pk_e, sk_e). This implicitly defines the message space $\mathfrak{M}_{gk}$, the ciphertext space $\mathfrak{C}_{gk}$ and the randomness space $\mathfrak{R}_{gk}$, where there must exist some public efficient injective mapping η satisfying $\eta(\mathbb{V}) \subset \mathfrak{M}_{gk}$. For sake of simplicity, we will assume that $\mathbb{V} \subset \mathfrak{M}_{gk}$ so that no encoding is needed. Finally, it chooses two groups X, Y and a one-way function $F : X \to Y$, which might depend on gk. The election public key pk is defined as $pk = (gk, \mathsf{crs}, pk_e, X, Y, F)$ and the election secret key is defined as $sk = sk_e$.

Register$(vid, rid, \mathbb{V}, pk)$**:** the algorithm generates the partial secret UCIV information $s_{uciv}^{vid,rid}$ for that voter by generating, for each voting option $v_i \in \mathbb{V}$, a random value $x_i^{vid,rid} \in X$. The partial public UCIV information $p_{uciv}^{vid,rid}$ is generated by computing the image $y_i^{vid,rid} = F(x_i^{vid,rid})$ for each voting option.

Vote$(vid, \sigma_v(s_{uciv}^{vid}), p_{uciv}^{vid}, pk, v)$**:** first, the algorithm parses pk as $(gk, \mathsf{crs}, pk_e, X, Y, F)$. It also parses $\sigma_v(s_{uciv}^{vid}) = \{\{x_i^{vid,rid}\}_{i \text{ s.t. } v_i \neq v}\}_{rid \in RID[vid]}$ and $p_{uciv}^{vid} = \{(y_1^{vid,rid}, \dots, y_n^{vid,rid})\}_{rid \in RID[vid]}$,where $RID[vid]$ is the set of registrars the voter registered with. The algorithm then computes $x_i^{vid} = \sum_{rid \in RID[vid]} x_i^{vid,rid}$ for i s.t. $v_i \neq v$ and $y_i^{vid} = \sum_{rid \in RID[vid]} y_i^{vid,rid}$ for $i \in \{1, \dots, n\}$. It then runs the Enc algorithm using randomness $r \in \mathfrak{R}_{gk}$, producing a ciphertext $c = \mathsf{Enc}(gk, pk_e, v, r)$. Then, by using the witness $w = (r, (\xi_1^{vid}, \dots, \xi^{vid}))$, where ξ_k^{vid} with $k = j$ is such that $v_j = v$ is the empty string ε and it is x_k^{vid} otherwise, the algorithm computes a NIZK-PoK for the relation $R_{2\text{-}cnf}$ as $\pi = \mathsf{Prove}(gk, \mathsf{crs}, c, y_1^{vid}, \dots, y_n^{vid}, w)$. The resulting ballot b is defined by the pair (c, π).

ValidateBallot(BB, b, vid)**:** upon reception of a ballot b, which can be parsed as $b = (c, \pi)$, it is checked if in the bulletin board there is another either another pair (id, b') (i.e., with the same id), or another ballot $b' = (c', \pi')$ such that $c = c'$. If any of such case is found, the algorithm stops and returns 0. Otherwise, p_{uciv}^{vid} is recovered from the bulletin board and parsed as $\{(y_1^{vid,rid}, \dots, y_n^{vid,rid})\}_{rid \in RID[vid]}$. Then, $y_i^{vid} = \sum_{rid \in RID[vid]} y_i^{vid,rid}$ for $i \in \{1, \dots, n\}$ is computed and VerifyProof$(gk, \mathsf{crs}, c$

$, y_1^{vid}, \dots, y_n^{vid}, \pi)$ is run, where gk, crs are also recovered from the bulletin board. The output of VerifyProof is returned as the output of ValidateBallot.

Tally(BB, sk)**:** at the end of the election, ValidateBallot(BB, b, vid) is run for all pairs (b, vid) appearing in the bulletin board. Then, each individual ballot b is decrypted $\tilde{v} = \mathsf{Dec}(gk, sk_e, c)$ and ρ is applied to the resulting decryptions $\{\tilde{v}\}$. The output of ρ is defined as the result and the proof of correct tabulation is defined to be the empty string ε.

As the proof of correct tabulation is the empty string ϵ, VerifyTally returns 1 on any input.

7.4.4 Extension to multiple-mark ballots

The protocol defined in Section 7.4.3 only applies to a specific type of election systems, those in which only one candidate can be chosen. We now extend the election systems for which our protocol works to multiple-mark ballot elections. Although we could consider $\mathbb{V}$ to be the space of the combinations of marks, a vote v to be a combination of marks and then use the protocol defined in Section 7.4.3 in a straightforward way, this would result in a terribly inefficient system. Therefore, we give a cleaner solution in this section.

If multiple candidates can be chosen, then multiple encryptions will be produced, one for each chosen candidate. Denote the number of encryptions by ℓ. We define the relation $R_{\ell-enc}$ as follows: $R_{\ell-enc} = \{(gk, (c_1, \dots, c_\ell, m), r) | c_1 = \mathsf{Enc}(gk, pk, m, r) \vee \cdots \vee c_\ell = \mathsf{Enc}(gk, pk, m, r)\}$. We then define the relation

$$\begin{aligned}
&R_{(\ell+1)\text{-}cnf} = \\
&\{(gk, ((c_1, \dots, c_\ell), (m_1, \dots, m_n), (y_1, ..., y_n)), (r, (x_1, \dots, x_n)))| \\
&\quad ((gk, (c_1, \dots, c_\ell, m_1), r) \in R_{\ell-enc} \vee (x_1, y_1) \in R_{ow}) \wedge \\
&\quad ((gk, (c_1, \dots, c_\ell, m_2), r) \in R_{\ell-enc} \vee (x_2, y_2) \in R_{ow}) \wedge \\
&\qquad\qquad\qquad \cdots \qquad\qquad\qquad\qquad \wedge \\
&\quad ((gk, (c_1, \dots, c_\ell, m_n), r) \in R_{\ell-enc} \vee (x_n, y_n) \in R_{ow})\}
\end{aligned}$$

The algorithms in the protocol remain almost the same as in the Core Voting Protocol exposed in Section 7.4.3. The Register algorithm will still generate a secret $x_i^{vid,rid}$ and a public $y_i^{vid,rid}$ value for each individual candidate ν_i, although the space $\mathbb{V}$ is now defined to contain all the valid combinations of candidates a voter can select.

The main difference is at the Vote algorithm, where the input v represents in this case a set $\{\nu_i\}_{i=1}^{\ell}$ of several candidates selected by the voter. Accordingly, the evaluation $\sigma_v(s_{uciv}^{vid})$ will provide the set of all secrets $\{x_i^{vid,rid}\}_{rid \in RID[vid]}$ except for the ones assigned to $\{\nu_i\}_{i=1}^{\ell}$.

During the execution of the Vote algorithm, one ciphertext is created for each of the ℓ candidates contained in v, and instead of generating the NIZK-PoK for the relation $R_{2\text{-}cnf}$, the NIZK-PoK for the relation $R_{(\ell+1)\text{-}cnf}$ is used.

The algorithms $\mathsf{ValidateBallot}$ and Tally also change accordingly: the algorithm $\mathsf{ValidateBallot}$ uses the NIZK-PoK verification algorithm for the same relation $R_{(\ell+1)\text{-}cnf}$; and the Tally process decrypts multiple ciphertexts per ballot. In addition, $\mathsf{ValidateBallot}$ checks that no individual ciphertext is found twice in the bulletin board.

As a final remark on multiple-mark ballots, the efficiency of a mix-net based protocol with several encryptions per voter is linear on the number of options encrypted, so the protocol will get less efficient as more marks can be made. In addition, the NIZK-PoK proofs will also be less space-efficient the more options a voter can choose.

7.5 Security of the protocol

In this Section we state that our single-mark protocol given in Section 7.4.3 satisfies ballot privacy, strong consistency, strong correctness and universal cast-as-intended verifiability. The security results are easily extended to the multiple-mark protocol of Section 7.4.4.

Theorem 7.5.1. *Let* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a NM-CPA encryption scheme and let* $(\mathsf{Setup}, \mathsf{GenCRS}, \mathsf{Prove}, \mathsf{VerifyProof})$ *be a NIZK-PoK for the relation* $R_{2\text{-}cnf}$ *defined in Section 7.4.2. Then, the protocol defined in Section 7.4.3 satisfies the ballot privacy property.*

Proof. Recall that the ballot privacy definition is based on the indistinguishability of two experiments which depend on a bit β. We will refer to them as Exp_0 for the experiment where $\beta = 0$ and Exp_1 for the experiment where $\beta = 1$.

Let $\mathsf{SimVote}(v, s^{id}_{uciv}, p^{id}_{uciv}, pk)$ be the Vote algorithm of the protocol given in Section 7.4.3 but, when computing the Prove algorithm of the NIZK-PoK, the witness w to be used is $w = (\epsilon, (\xi^{vid}_1, \ldots, \xi^{vid}_n))$, where ξ^{vid}_k is x^{vid}_k, the pre-image of y^{vid}_k for all k. Note that w is a valid witness and $\mathsf{SimVote}$ is defined to take as input *all* values x^{vid}_k. Then consider the experiment $\mathsf{Exp}_{\beta'}$ which is the same as Exp_β except that, when the adversary submits a **Vote**(id, v_L, v_R) query, the challenger runs $\mathsf{SimVote}(v_L, s^{id}_{uciv}, p^{id}_{uciv}, pk)$ and $\mathsf{SimVote}(v_R, s^{id}_{uciv}, p^{id}_{uciv}, pk)$ instead of $\mathsf{Vote}(\sigma_{v_L}(s^{id}_{uciv}), p^{id}_{uciv}, pk, v_L)$ and $\mathsf{Vote}(\sigma_{v_R}(s^{id}_{uciv}), p^{id}_{uciv}, pk, v_R)$ respectively. The following lemma is straightforward to prove.

Lemma 7.5.2. *The experiments* Exp_β *and* $\mathsf{Exp}_{\beta'}$ *are computationally indistinguishable for* $\beta \in \{0, 1\}$ *if the NIZK-PoK is computationally witness indistinguishable.*

Now consider the $\mathsf{Enc2Vote}$ scheme as defined in Section 5.4. In addition, consider the definition of ballot privacy given in Section 7.3.2 in the special case that $SVI, PVI = \emptyset$ (this is, no UCIV information and therefore functionality is used). The resulting definition is the same than the one given in Section 5.3.3. As mentioned in Section 5.4, the following theorem holds.

Theorem 7.5.3. *Let* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a NM-CPA secure encryption scheme, then* Enc2Vote *has ballot privacy, for the ballot privacy definition in Section 5.3.3.*

Finally, we reduce the ballot privacy property of our scheme to the ballot privacy property of the Enc2Vote scheme.

Lemma 7.5.4. *Let* $\mathcal{A}^1$ *be a p.p.t. adversary that interacts with a challenger* $\mathcal{C}$ *and outputs a bit* $\alpha^{\mathcal{A}}$ *such that* $|\Pr[\alpha^{\mathcal{A}}|\mathsf{Exp}_{0'}] - \Pr[\alpha^{\mathcal{A}}|\mathsf{Exp}_{1'}]|$ *is non-negligible. Then, there exists an adversary* $\mathcal{A}^{\mathsf{E2V}}$ *that breaks the ballot privacy property of the* Enc2Vote *scheme.*

In our reduction, $\mathcal{A}^1$ will interact with $\mathcal{A}^{\mathsf{E2V}}$ (the latter acting as the challenger) and $\mathcal{A}^{\mathsf{E2V}}$ will interact with a challenger $\mathcal{C}$.

The reduction is as follows:

In the **Setup** phase, $\mathcal{C}$ will run the Setup and KeyGen algorithms, keeping the private key for itself and publishing the public key pk_{E2V} to the bulletin board. Then, $\mathcal{A}^{\mathsf{E2V}}$ will run the GenCRS algorithm and will post the public key $pk_{uciv} = (pk_{\mathsf{E2V}}, gk, \mathsf{crs})$ to its bulletin board.

In the **Registration** phase, when $\mathcal{A}^1$ submits a **Register** query, $\mathcal{A}^{\mathsf{E2V}}$ will run the Register algorithm of our protocol, it will give all the output to $\mathcal{A}^1$, it will also keep it for itself and will post the partial public UCIV information to the bulletin board.

In the **Voting** phase when $\mathcal{A}^1$ submits a **Vote** query, $\mathcal{A}^{\mathsf{E2V}}$ will submit a **Vote** query to $\mathcal{C}$. The challenger will respond by publishing a ballot $b_{\mathsf{E2V}} = c$ to the bulletin board visible by $\mathcal{A}^{\mathsf{E2V}}$. Then, $\mathcal{A}^{\mathsf{E2V}}$ will run $\mathsf{Prove}(gk, \mathsf{crs}, c, y_1^{vid}, \ldots, y_n^{vid}, w)$ with $w = (\epsilon, (x_1^{vid}, \ldots, x_n^{vid}))$, where $x_1^{vid}, \ldots, x_n^{vid}$ and $y_1^{vid}, \ldots, y_n^{vid}$ are computed as in our voting protocol. Let π be the output of the Prove algorithm. $\mathcal{A}^{\mathsf{E2V}}$ will then post the ballot $b_{priv-uciv} = (c, \pi)$ to the bulletin board visible by $\mathcal{A}^1$. Finally, when $\mathcal{A}^1$ submits a **Ballot**(b, id) query, where $b = (c, \pi)$, $\mathcal{A}^{\mathsf{E2V}}$ will construct a ballot $\tilde{b} = c$ and will submit a **Ballot**$(\tilde{b}, id)$ query to $\mathcal{C}$.

In the **Tallying** phase, the challenger $\mathcal{C}$ will post the result of evaluating $\mathsf{Tally}(L, sk_e)$ to the bulletin board visible by $\mathcal{A}^{\mathsf{E2V}}$. Then, $\mathcal{A}^{\mathsf{E2V}}$ will publish the same result to the bulletin board visible by $\mathcal{A}^1$. We set the correct tabulation proofs (both the real one and the simulated one) to be the empty string ε.

Finally, $\mathcal{A}^1$ will output a bit and $\mathcal{A}^{\mathsf{E2V}}$ will output the same bit as its own output.

It is straightforward to see that the output of $\mathcal{A}^{\mathsf{E2V}}$ in its interaction with $\mathcal{A}^1$ is correctly distributed, which implies that the reduction is sound. □

Recall that, as explained in Section 5.3.4, in order to have a meaningful notion of privacy we also need that the voting protocol satisfies the strong consistency and strong correctness properties.

Theorem 7.5.5. *Let* ρ *be the counting function which outputs its inputs randomly permuted and filtering invalid votes. Let* Extract *be the algorithm defined as* $\mathsf{Extract}(b, sk) = \mathsf{Dec}(gk, sk_e, c)$. *Then, the protocol defined in Section 7.4.3 has strong consistency with respect to* Extract.

Theorem 7.5.6. *The protocol defined in Section 7.4.3 has strong correctness.*

Both theorems can be reduced to proving that the Enc2Vote satisfies those properties. The reductions are very similar to the one used to prove Lemma 7.5.4. On the other hand, Enc2Vote does satisfy strong correctness and strong consistency with respect to Extract as explained in Section 5.4. Therefore, our voting protocol also satisfies those properties.

Let us now prove that our scheme satisfies the universal cast-as-intended property. However, we have to limit to the case where the number of voters and candidates are polynomial in the security parameter λ (*In the security reduction we guess which voter identity and which voting option the adversary will try to attack. This implies that the security reduction will not be tight. This loss of tightness can be avoided if we assume that the adversary announces in advance which voter and voting option it will attack*).

Theorem 7.5.7. *Let $F : X \to Y$ be a homomorphic one-way function and let* $(\mathsf{Setup}, \mathsf{GenCRS}, \mathsf{Prove}, \mathsf{VerifyProof})$ *be a NIZK-PoK. Let $\mathcal{A}^{uciv}$ be an adversary which makes* ***Register*** *queries for at most a polynomial number (in the security parameter) of different voter identities. Also, assume that the number of candidates is polynomial in the security parameter. Then, the protocol defined in Section 7.4.3 satisfies the universal cast-as-intended property.*

Proof. In this proof, we assume that there is an adversary $\mathcal{A}^{uciv}$ which breaks the universal cast-as-intended property and we will construct an adversary $\mathcal{A}^{owf}$ which will break the one-wayness of F. $\mathcal{A}^{owf}$ will make use of the extraction property of the NIZK-PoK during the security reduction.

First, $\mathcal{A}^{owf}$ will guess which will be the voter identity which $\mathcal{A}^{uciv}$ will attack. Concretely, it will guess that it will be the k-th queried identity, where k will be drawn from a uniform distribution between 1 and q. Similarly, it will guess that the $\mathcal{A}^{uciv}$ will target the vote v_ℓ, where v will be drawn from a uniform distribution between 1 and n. Note that this sets the probability of correctly guessing which voter and voting option the adversary $\mathcal{A}^{uciv}$ to be $1/(qn)$.

In the **Setup**, $\mathcal{A}^{owf}$ will run the algorithm ExtGenCRS of the zero-knowledge extractor instead of running GenCRS to output a common reference string crs and a trapdoor key tk. It will keep tk for itself and will define pk, sk as in the protocol.

In case of **Register** queries, $\mathcal{A}^{uciv}$ submits one voter identity vid and one registrar identity rid. We distinguish between two cases:

- If (i) vid is the k-th voter identity which $\mathcal{A}^{uciv}$ is registering, (ii) $rid \notin \widetilde{RID}$ and (iii) rid is the first registrar not in $\widetilde{RID}$ with which vid is being registered then $\mathcal{A}^{owf}$ computes the partial public and secret UCIV information by running Register but redefines $y_\ell^{vid,rid} = y$ the one-way challenge and implicitly defines $x_\ell^{vid,rid}$ to be its pre-image. We denote rid* the registrar rid.
- Otherwise, $\mathcal{A}^{owf}$ computes the partial public and secret UCIV information by running Register and, if $rid \in \widetilde{RID}$ it forwards all the information to the adversary.

Note that if the guess of which vid and voting option is going to be attacked is correct then $\mathcal{A}^{owf}$ will be able to answer to any **CorruptVotingDevice** or **CorruptVoter** queries. This happens with probability $1/(qn)$.

In this case, once $\mathcal{A}^{uciv}$ has submitted the pair (b^*, vid^*) where $b^* = (c^*, \pi^*)$, $\mathcal{A}^{owf}$ will use the algorithm $\mathsf{Extract}$ of the NIZK-PoK using the trapdoor key tk to obtain a witness w for π^*. Due to the structure of the NIZK-PoK language, it must be the case that the witness w has the form $w = (r, (x_1^{vid}, \ldots, x_n^{vid}))$ with $y_\ell^{vid} = F(x_\ell^{vid})$.

Denote RID^* the set $RID[vid^*]\backslash rid^*$. Then, by definition of y_ℓ^{vid} and x_ℓ^{vid} we have that $x := x_\ell^{vid} - \sum_{rid \in RID^*} x_\ell^{vid,rid}$ is a valid pre-image of $y := y_\ell^{vid,rid^*}$. Finally, observe that if the advantage Adv of $\mathcal{A}^{uciv}$ is negligible, then the advantage of $\mathcal{A}^{owf}$ which is $1/(nq) \cdot \mathsf{Adv}$ will also be negligible. □

7.6 Protocol instantiations

7.6.1 Efficient instantiation with ROM-based security

The protocol given in Section 7.4 uses a generic encryption scheme and a generic (and maybe impractical) NIZK-PoK. In this section we give a concrete instantiation of the voting protocol. The instantiation has provable security in the Random Oracle Model (ROM).

This instantiation of our protocol uses the Signed ElGamal encryption scheme, which is NM-CPA secure in the ROM, together with σ-protocols made non-interactive by using the Fiat-Shamir heuristic, which results in a NIZK-PoK in the ROM. We now present the mentioned building blocks.

Signed ElGamal encryption

The encryption scheme that we use is the Signed ElGamal cryptosystem, as defined in Section 2.4.1. The Signed ElGamal encryption scheme is NM-CPA secure in the ROM assuming that the DDH problem is hard in the ElGamal group $\bar{\mathbb{G}}$.

Discrete logarithm one-way function

The one-way function that we will use is the exponentiation function introduced in Section 2.6. By using the same cyclic group $\bar{\mathbb{G}}$ and the same generator $\bar{g}$ used for the encryption scheme, the exponentiation function $F : \mathbb{Z}_p \to \bar{\mathbb{G}}$ is defined as $F(x) = x\bar{g}$. The exponentiation function is homomorphic and, in addition, this function is one-way if the discrete logarithm problem is hard in $\bar{\mathbb{G}}$. Note that the discrete logarithm assumption is a weaker assumption than the DDH assumption, therefore we are not introducing any new assumption.

NIZK-PoK from σ-protocols

As we explained in Section 2.7.2, one way to build NIZK-PoKs with ROM-based security is to apply the Fiat-Shamir heuristic to σ-protocols. There exist σ-protocols for a wide range of languages. Two examples are proving knowledge of a discrete logarithm [SJ00] (also known as a Schnorr protocol) and proving equality of two discrete logarithms [CP92] (also known as a Chaum-Pedersen protocol). Furthermore, as shown in [CDS94], given a σ-protocol for a relation R_1 and another σ-protocol for a relation R_2 one can construct both a σ-protocol for the relation R_{or} defined as $(x, w) \in R_{or}$ iff $(x, w) \in R_1$ or $(x, w) \in R_2$ and a σ-protocol for the relation R_{and} defined as $(x, w) \in R_{and}$ iff $(x, w) \in R_1$ and $(x, w) \in R_2$.

Focusing on the election scheme for single-mark ballots with n candidates, a ballot in this specific instantiation of our scheme will have the form (C, π), where $C = (\bar{c}_1, \bar{c}_2, c_3, c_4)$ is a Signed ElGamal encryption of the chosen candidate $\bar{v}$, and $\pi = (w, \{w_1^{(i)}, s_1^{(i)}, s_2^{(i)}\}_{1\leq i\leq n})$ is the NIZK-PoK for the relation $R_{2\text{-}cnf}$, computed by combining the Fiat-Shamir heuristic and the afore-mentioned techniques, as we now detail. Let $C' = (\bar{c}_1, \bar{c}_2) = (r\bar{g}, r\bar{h} + v)$ be the underlying ElGamal encryption of C and let us denote $x_i = x_i^{vid,rid}$ and $y_i = y_i^{vid,rid}$, for each $i = 1, \ldots, n$. Then, the proof π can be computed as follows:

1. For $i = 1, \ldots, n$, $i \neq j$, choose $s_1^{(i)}, w_1^{(i)} \in \mathbb{Z}_p$ at random. Define the values $\bar{c}_1^{(i)} = s_1^{(i)}\bar{g} - w_1^{(i)}\bar{c}_1$ and $\bar{c}_2^{(i)} = s_1^{(i)}\bar{h} + (-w_1^{(i)})(\bar{c}_2 - \bar{v}_i)$.
2. For index j, choose $s_2^{(j)}, w_2^{(j)} \in \mathbb{Z}_p$ at random, and define the value $\bar{r}^{(j)} = s_2^{(j)}\bar{g} - w_2^{(j)}\bar{y}_j$.
3. For $i = 1, \ldots, n$, $i \neq j$, choose $\alpha^{(i)} \in \mathbb{Z}_p$ at random and compute $\bar{r}^{(i)} = \alpha^{(i)}\bar{g}$.
4. For index j, choose $\beta^{(j)} \in \mathbb{Z}_p$ and $z^{(j)} \in \bar{\mathbb{G}}$, at random, and compute $\bar{c}_1^{(j)} = \beta^{(j)}\bar{g}$ and $\bar{c}_2^{(j)} = \beta^{(j)}\bar{h}$.
5. Compute the hash function $w = H(\bar{c}_1, \bar{c}_2, c_3, c_4, \{\bar{c}_1^{(i)}, \bar{c}_2^{(i)}, \bar{r}^{(i)}\}_{1\leq i\leq n}) \in \mathbb{Z}_p$.
6. For $i = 1, \ldots, n$, $i \neq j$, compute $w_2^{(i)} = w - w_1^{(i)} \mod p$. For index j, compute $w_1^{(j)} = w - w_2^{(j)} \mod p$.
7. For $i = 1, \ldots, n$, $i \neq j$, compute $s_2^{(i)} = \alpha^{(i)} + x_i w_2^{(i)} \mod p$.
8. For index j, compute $s_1^{(j)} = \beta^{(j)} + r w_1^{(j)} \mod p$.

To verify the correctness of the proof π, one has to compute first, for each $i = 1, \ldots, n$, the values $w_2^{(i)} = w - w_1^{(i)} \mod p$, $\bar{c}_1^{(i)} = s_1^{(i)}\bar{g} - w_1^{(i)}\bar{c}_1$, $\bar{c}_2^{(i)} = s_1^{(i)}\bar{h} - w_1^{(i)}(\bar{c}_2 - \bar{v}_i)$ and $\bar{r}^{(i)} = s_2^{(i)}\bar{g} - w_2^{(i)}\bar{y}_i$. After that, the proof is accepted if and only if $w = H(\bar{c}_1, \bar{c}_2, c_3, c_4, \{\bar{c}_1^{(i)}, \bar{c}_2^{(i)}, \bar{r}^{(i)}\}_{1\leq i\leq n})$.

Finally, note that we are computing the NIZK-PoK on the ElGamal ciphertext (not in the Signed ElGamal). In other words, the voter is assured that the ElGamal ciphertext contains the selected candidate. However, as the Schnorr proof of the Signed ElGamal encryption is publicly verifiable, this assures the voter that the Signed ElGamal ciphertext contains the selected candidate too. Moreover, the ciphertext which consists of the Signed ElGamal ciphertext and the NIZK-PoK is $\mathsf{NM\text{-}CPA}$ secure even if the NIZK-PoK is computed on the ElGamal ciphertext.

Efficiency comparison to Helios

The group $\bar{\mathbb{G}}$ is typically implemented with an elliptic curve [Mil85]. In this case, each element in $\mathbb{Z}_p$ and each element in $\bar{\mathbb{G}}$ can be represented with λ bits, where λ is the length in bits of the prime number p. Therefore, the size of each ballot (c, π) in this instantiation is $(3n + 5)\lambda$ bits: 4λ bits for c and $(3n + 1)\lambda$ bits for π. The number of modular exponentiations that must be computed to generate such a ballot is $3n+4$.

We can compare these costs with the costs of the basic version of Helios for the case of a single-mark election, with n candidates. Let us remember that a ballot in Helios consists of n ElGamal encryptions $c_1, \ldots, c_n$ of 0 or 1, one encryption for each of the n candidates (the voter encrypts 1 only for the chosen candidate, and 0 everywhere else), along with a NIZK-PoK π of the fact that each ciphertext encrypts 0 or 1, and the product of all the ciphertexts (which is an encryption of the sum of the n plaintexts, by the homomorphic properties of ElGamal) also encrypts 0 or 1. The size of each ballot in Helios is $(5n + 4)\lambda$ bits: $2n\lambda$ bits for the n ElGamal ciphertexts, and $(3(n + 1) + 1)\lambda$ bits for π. The number of modular exponentiations required to generate a ballot in Helios is $6n$.

Summing up, the instantiation that we have just presented is more efficient than the basic version of Helios, for single-mark elections, in terms of the size of the ballots and the computational cost to generate them, while adding a new property (universal cast-as-intended verifiability) not satisfied by Helios.

Even though Helios uses homomorphic tallying, resulting in a much cheaper tallying process than a Mix-Net based protocol, this efficiency comparison shows that our new protocol has ballots of a reasonable size and that ballot creation is efficient enough to be used in a real election.

7.6.2 Instantiation with security in the CRS model

As we mentioned in Section 2.3, the security obtained when instantiating a random oracle by a hash function is heuristic, which led the Random Oracle Model to some criticisms. Therefore, at least from a theoretical point of view, it is interesting to obtain protocols which security is not based on the Random Oracle Model. The Common Reference String model states that there might be a common reference string which is honestly created and trusted by all parties prior to the protocol. We still want the protocol to be *efficient* and not just a theoretical result. Therefore, we will need to choose carefully the building blocks of out voting protocol.

Setup

The setup is a type-II bilinear group $(p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathcal{PG}_{type-II}$.

Cramer-Shoup encryption scheme

To achieve security in the common reference string model we can use the DLIN-based version of the Cramer-Shoup encryption scheme, introduced in Section 3.5.1, in $\check{\mathbb{H}}$, which is an $\mathsf{IND\text{-}CCA}$ secure encryption scheme if the DLIN assumption holds in $\check{\mathbb{H}}$. This will be used to encrypt the selection $\check{m}$, which will be one of the candidates $\{\check{v}_i\}_{i=1}^n \in \check{\mathbb{H}}$.

One-way function in the bilinear group

Although one would be tempted to use the exponentiation function in the groups $\hat{\mathbb{G}}$ or $\check{\mathbb{H}}$, it turns out that then it would not be easy to construct a NIZK-PoK for the relation $R_{2\text{-}cnf}$ defined in Section 7.4.2 in the CRS model. We therefore use a different one-way function. In particular, the function that we use is the map $\psi : \check{\mathbb{H}} \to \hat{\mathbb{G}}$, which is homomorphic and, in addition, it is also one-way if the bilinear group is of type-II.

Groth-Sahai proofs

Groth-Sahai proofs, introduced in Section 2.8 and re-framed as a Commit-and-Prove scheme as shown in Section 4, are the most flexible and efficient NIZK-PoK protocols secure in the Common Reference String model. As seen in Section 4, Groth-Sahai proofs (possibly re-framed as a Commit-and-Proof scheme) allow us to build proofs of satisfiability of equations of the form

$$\sum_{i=1}^{n} e(\hat{a}_i, \check{y}_i) + \sum_{i=1}^{m} e(\hat{x}_i, \check{b}_i) + \sum_{i=1}^{m}\sum_{j=1}^{n} \gamma_{ij} e(\hat{x}_i, \check{y}_j) = e(\hat{g}, \check{d}) + e(\hat{c}, \check{h}), \tag{7.1}$$

where $\{\hat{x}_i\}_{i=1}^m$ are variables in $\hat{\mathbb{G}}$, $\{\check{y}_i\}_{i=1}^n$ are variables in $\check{\mathbb{H}}$, $\{\hat{a}_i\}_{i=1}^n, \hat{c}$ are constants in $\hat{\mathbb{G}}$, $\{\check{b}_i\}_{i=1}^m, \check{d}$ are constants in $\check{\mathbb{H}}$ and $\{\gamma_{i,j}\}_{i,j=1}^{m,n} \in \mathbb{Z}_q$ are also constants. A witness for a statement is a set of values of $\hat{x}_i, \check{y}_j$ such that the set of equations is satisfied.

We can encode the relation $R_{2\text{-}cnf}$ defined in Section 7.4.2 as a set of equations of the form (7.1). A Groth-Sahai proof of satisfiability of such set of equations is a NIZK-PoK for $R_{2\text{-}cnf}$ if the techniques shown in Section 4 are used. Indeed, observe that equations of the form (7.1) only contain group elements as variables, which gives us perfect extractability.

Let $\hat{m}$ be a variable in $\hat{\mathbb{G}}$ (which represents the plaintext of the encrypted selection and is computed as $\psi(\check{m})$), $\{\hat{v}_i\}_{i=1}^n$ be constant elements in $\hat{\mathbb{G}}$ (which represent the possible candidates and are computed as $\psi(\check{v}_i)$), $\{\check{y}_i\}_{i=1}^n$ be variables in $\check{\mathbb{H}}$ (which represent the pre-image values), $\{\hat{f}_i\}_{i=1}^n$ be constants in $\hat{\mathbb{G}}$ (which represent the image values), $\{\hat{z}_i\}_{i=1}^n$ be variables in $\hat{\mathbb{G}}$, $\{\check{w}_i\}_{i=1}^n$ be variables and $\check{\mathbb{H}}$ ($\hat{z}_i, \check{w}_i$ are auxiliary variables) and $\hat{g}, \check{h}$ be the generators of $\hat{\mathbb{G}}, \check{\mathbb{H}}$ respectively. Then, the set of equations which encodes the relation $R_{2\text{-}cnf}$ is the following set of $4n$ equations:

$$e(\hat{m}, \check{w}_i) = e(\hat{v}_i, \check{w}_i) \qquad \text{for } i \in \{1, \ldots, n\} \tag{1-i}$$

$$e(\hat{g}, \check{y}_i) = e(\hat{f}_i, \check{h} - \check{w}_i) \qquad \text{for } i \in \{1, \ldots, n\} \tag{2-i}$$

$$e(\hat{z}_i, \check{h}) = e(\hat{g}, \check{w}_i) \qquad \text{for } i \in \{1, \ldots, n\} \tag{3-i}$$

$$e(\hat{z}_i, \check{h}) = e(\hat{z}_i, \check{w}_i) \qquad \text{for } i \in \{1, \ldots, n\} \tag{4-i}$$

It is easy to see that, by moving all non-constant terms to the left hand side of the equations and the constant terms to the right hand side of the equations, the equations above are indeed of the form 7.1.

A straightforward analysis shows that, for a fixed i, equations (3-i) and (4-i) are only satisfied either if $\hat{z}_i = \hat{g}, \check{w}_i = \check{h}$ or if $\hat{z}_i = \hat{0}, \check{w}_i = \check{0}$, where $\hat{0}$ and $\check{0}$ denote the identity elements in $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$ respectively. This is what allows us to have an *or*-proof. Indeed, if $\check{w}_i = \check{h}$, then equation (1-i) is only satisfied if $\hat{m} = \hat{v}_i$, while equation (2-i) is trivially satisfied for $\check{y}_i = \check{0}$. On the other hand, if $\check{w}_i = \check{0}$, then equation (1-i) is satisfied for any value of $\hat{m}$, while equation (2-i) is only satisfied if $\check{y}_i$ is such that $\psi(\check{y}_i) = \hat{f}_i$. In other words, for a fixed i, the set of equations (1-i)-(4-i) correspond to the statement "either m (the value encrypted in the ciphertext) is equal to v_i, or I know a pre-image of f_i". Therefore, the set of equations $\{$(1-i),(2-i),(3-i),(4-i)$\}_{i=1}^{n}$ correctly encodes the relation $R_{2\text{-}cnf}$.

Let us now consider the Commit-and-Prove scheme given in Section 4 as the proof system used to show satisfiability of the above system of equations. Even though the protocol given in Section 4 is designed for type-III bilinear groups, as explained in Section 4.10 they can easily be adapted to the type-II bilinear group setting given in [GSW10].

In this case, we can define the type of $\hat{m}$ to be encryption, whereas we can use the type commitment for all other variables. As shown in Figure 4.6, this combination of types is an acceptable one.

Finally, observe that the number of elements in both $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$ is $O(n)$, and the constant is relatively small, which translates into a fairly efficient instantiation.

The last thing we need to do is to prove that the Cramer-Shoup encryption of $\check{m}$ contains the *same* voting option as the encryption of $\hat{m}$ given in the Commit-and-Prove proof explained above.

This can be solved by using the techniques given in Section 3.5.3 to prove plaintext equality. Even though the techniques given in Section 3.5.3 apply to the symmetric case, they can be adapted to work in an asymmetric setting where the commitment keys to $\hat{m}$ are the commitment keys of the Commit-and-Prove scheme and the commitment keys to $\check{m}$ are the Cramer-Shoup tailored commitment keys explained in Section 3.5.1.

The proof is relatively small and only adds a constant number of group elements to the encrypted vote sent by the voter.

7.7 Towards designing usable UCIV systems

The protocol presented in Section 7.4 requires the voter to provide one secret value per registrar and per each of the candidates that she did not choose to the voting device. This may require a considerable effort, since the number of available candidates in an election may be quite large, even if the voter can only make one selection (as in single-marking ballot elections). In addition, these values will be fairly big, for instance 256 bits if elliptic curves are used. Finally, providing secrets for the non-selected candidates is counter-intuitive (note that it is just inverse to what is required in code voting schemes [MMP02], for example), since the voter usually expects to enter information related to what she selects. Clearly, requiring the voter to introduce such information is not very user-friendly.

In this section, we present two voting systems built on top of our new voting protocol which provide a more intuitive approach for the voter introducing the codes. We provide these examples as a first step towards an implementation in a real scenario: usability tests should be done to assess how usable these systems really are.

One characteristic that we will see is that there is a trade-off between the number of registrars a voter registers with and the usability of the scheme. For example, in the first system the voter will need to use as many hardware tokens as registrars she registered with. However, this trade-off seems inevitable: the less trust there is in registrars the more security mechanisms we will need to provide.

7.7.1 Voting system using a cryptographic hardware token

The first system we propose is based on using a cryptographic hardware token, such as a Smart Card. This hardware token will be used to provide the secrets generated in the $\mathsf{Registration}$ phase to the voter. The cryptographic hardware token can be seen as an implementation of the secure channel between a registrar and a voter, which can not be eavesdropped by the voting device. In addition, the token is also used to provide some logic capabilities, so that the voter is only required to provide secrets for the voting option that she did indeed choose.

In order to do that, for each registrar the secret values for the n candidates available in the election are stored in a token (one token per registrar) in the following way: a random short password (i.e., a PIN) is generated for each of the candidates. Then, a $Reveal$ functionality is implemented in the token which, given the ℓ PINs corresponding to the candidates chosen by the voter, reveals the set of partial secret UCIV information corresponding to the candidates not chosen by the voter. Note that the set of PINs for a token must not be known to the voting device, so they have to be provided to the voter using a channel which can not be eavesdropped by the voting device.

This system is specially suitable for, but not restricted to, poll-site or kiosk-based electronic voting, where specific hardware such as cryptographic hardware tokens can be easily provided to the voters.

7.7.2 Voting system using QR codes

The second voting system that we propose is based on using high-capacity barcodes such as QR codes [ISO06]. These QR codes will contain the secret values the voter has to enter in the voting device. A paper sheet with printed QR codes provides means for storage for those secrets, as well as a channel which can not be eavesdropped by the voting device. The security of the channel depends on the method for delivering the QR sheets to the voters, for example in sealed envelopes sent by postal mail.

In contrast to the cryptographic hardware token, the QR sheet does not provide logic capabilities, and thus the $Reveal$ functionality is not present. However, the QR contents can still be organized in a way such that the voter is only required to scan the QR codes corresponding to the candidates she selects in order to retrieve the secrets corresponding to the other candidates.

For each registrar, a QR paper sheet will be created as follows. First, one QR code will be assigned to each candidate. Let n be the total number of candidates and let ℓ be the number of candidates which the voter can select. Then, each secret value $x_i^{vid,rid}$ (which corresponds to the candidate v_i) will be divided into $n-1$ shares using a threshold secret sharing scheme (defined in Section 2.6) with threshold ℓ. This will result in shares $x_{i,k}^{vid,rid}$ for $k \in \{1, \dots, n\} \backslash i$, this is, $n-1$ shares, each one assigned to each candidate different from v_i. Then, the QR code assigned to the candidate v_i will be created containing all the shares $x_{j,i}^{vid,rid}$ for $j \in \{1, \dots, n\} \backslash i$, this is, the i-th share of each secret value except for $x_i^{vid,rid}$.

It follows that, given the information of ℓ QRs codes (assigned to ℓ candidates), only the secret values for the candidates not assigned to those QR codes can be reconstructed from the shares inside the QR codes. Scrath surfaces or stickers can be used in order to enforce that only the designated QR codes are scanned by the voting device's camera.

Efficiency considerations need to be done when considering usage of QR codes: the information that they can contain is limited. Note however that each QR code will contain $n-1$ secret values of size 256 bits each (if using elliptic curves). The maximum capacity of a QR code is around 23 thousand bits, more than enough to be able to use them in most elections.

This system can be used either in remote or in poll-site/kiosk-based electronic voting schemes. The QR sheets may be distributed to the voters by postal mail or by hand, depending on the scenario.

7.7.3 Fully auditable voting systems

We have presented two different voting systems based on our new voting protocol. These systems have different mechanisms for delivering the secret values to the voters which can be used to easily provide these secret values to the voting devices. These voting systems significantly improve the usability of the core protocol from Section 7.4.

The strength of the cast-as-intended verification depends on the right secret values being provided by the voter to the voting device to compute the proof, and therefore it is critical that (i) the secret values and the corresponding images are computed correctly, and (ii) the mechanism to provide secret values to the voting device works as intended, i.e., it only provides the secret values which are supposed to be provided.

As shown in Section 7.3.2, the privacy of the encrypted vote does not depend on the knowledge of the secret values associated to the candidates. In particular, after the vote has been cast they can be made public, these secret values can be published.

When considering our two voting systems, this implies that the hardware tokens and the QR code sheets can be given to auditors after the election takes place without affecting voters' privacy. In addition, as long as the voting device does not get knowledge of the secret values associated to the candidates, both the hardware token and the QR code sheet can be audited before casting the vote. In order to audit the QR code sheet, all the QR codes can be read and it can be checked that they contain the correct shares. On the other hand, to audit the hardware token, a dump of its functionalities can be made.

Third-party auditors, international observers, and representatives of the different political parties and of the electoral commission should participate in this audit, as well as in the audit of the proofs generated for the cast votes.

It is important to note that our system allows to verify all the tokens or QR code sheets used by the voters. This contrasts with other verifiable voting systems [CRS05], [CCC$^+$09], where voting and/or verification materials cannot be used by the voters once they are audited and, therefore, only a sample of them is verified for correctness.

7.8 Conclusions and future work

In this work, we have proposed the first solution for the problem of universal cast-as-intended verification in electronic voting: in our solution everybody (and not only the voter, as it happens in previous solutions) is able to verify that the contents of an encrypted vote are consistent with the voter choices.

One obvious weakness of our solution is that its security only holds in front of honest-but-curious registrars. Removing this restriction is not an easy task. This is because that voters, unlike voting devices, can not perform cryptographic computations. Therefore, a voter has no means to tell whether the partial secret UCIV information that he receives from a registrar matches the partial public UCIV information, even if the registrars use zero-knowledge proofs or digital signatures.

Intuitively, a solution would be to be able to blame registrars who behave incorrectly. This blame could serve two purposes. On one hand, it could help a voter to identify which registrars she can trust. On the other hand, it could be useful to determine that a registrar is malicious and therefore should be excluded from the election. In the latter case, voters should not be able to forge false claims, a task which seems difficult to solve at the protocol level.

In any case, proper security definitions should be given to prevent both voters making bogus claims but also registrars from creating incorrect UCIV information in a way that no one can prove their incorrect behavior.

Another interesting line of research would be to improve the efficiency of the voting protocol. In particular, the size of ballots scales poorly with the number of candidates and the number of choices a voter can make. This is in contrast with typical (non-UCIV) mix-net protocols, where the size of ballots increases at most linearly with the number of choices a voter can make.

Although in this work we focused on the notions of privacy provided by existing schemes such as [Adi08], we consider for future works to design a protocol with universal cast-as-intended verifiability, together with stronger notions of privacy such as receipt-freeness or coercion-resistance. Another challenge to solve is to design a voting protocol such that the voting device does not learn the identity of the voter even in front of dishonest registrars.

Finally, we consider there is work to do in the usability field. Although we have proposed solutions for making the voting process more intuitive for the voter, there is still a long way to walk to have a usable voting scheme. Feedback from usability tests and user experience designers should be used in order to find out the best approaches to follow.

Chapter 8

Conclusions and Open Problems

Regarding our contributions in cryptography, we have shown how to improve known Groth-Sahai proofs in several directions, either by changing the underlying assumptions, changing the generic Groth-Sahai proofs construction or fine-tuning the proofs for concrete statements. Our improvements directly benefit many protocols which use Groth-Sahai proofs in a black-box way. The question therefore is whether we can push the efficiency of Groth-Sahai proofs even further without changing their security properties, since each improvement, even small ones, benefits many cryptographic protocols. Another question is which is the best efficiency we can get by improving Groth-Sahai proofs. Answering this question would allow us to know how efficient protocols based on Groth-Sahai proofs can theoretically get just by leveraging the improvements in Groth-Sahai proofs.

The contributions of our work in electronic voting are of different nature. We have focused on solving problems which have a direct impact on the deployment of electronic voting systems. We have shown how to make electronic voting systems robust to spoiled votes, not from an accuracy or privacy point of view but from a reputation point of view. In our work, we have proved that vote validatability is an achievable security property by building an electronic voting protocol, based on bilinear groups, satisfying vote validatability. However, we would like to see whether currently used electronic voting protocols can be adapted to satisfy vote validatability without relying on bilinear groups.

We have also demystified the perception that cast-as-intended verifiability can only be achieved by implementing individual verifiability mechanisms. This is an important change of paradigm since it removes part of the verifiability burden from the voter, therefore achieving more usable electronic voting systems. We already stated some of the open problems on the conclusions of that contribution, such as building electronic voting protocols which are universally cast-as-intended verifiable in presence of malicious registrars which are not limited to be honest-but-curious. In addition to those conclusions, it is still an open question if we can remove or minimize individual verifiability from end-to-end verifiable electronic voting systems and, in particular, if it could be possible to design recorded-as-cast mechanisms which are universally verifiable.

The last conclusion we would like to point out is rather philosophical. In this thesis, we have presented some contributions in cryptography and others in electronic voting. However, these contributions are not independent: we have used results from our cryptographic contributions to instantiate electronic voting protocols with new security properties. This puts into evidence the fact that there is a strong connection between the state-of-the-art in cryptographic protocols and electronic voting protocols. Even if some cryptography research seems rather theoretical, it might eventually lead to practical applications in areas such as secure electronic voting. In order for electronic voting research to benefit from cryptography advancements it is important for electronic voting researchers to have a deep knowledge of the latest developments in cryptography research. In summary, it is only when practical and theoretical research go hand-by-hand that advances in real-world problems happen.

Bibliography

[Adi08] Ben Adida. Helios: Web-based open-audit voting. In Paul C. van Oorschot, editor, *Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA*, pages 335–348. USENIX Association, 2008.

[ADR02] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer, 2002.

[AFG$^+$16] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. *J. Cryptology*, 29(2):363–421, 2016.

[AGOT14] Masayuki Abe, Jens Groth, Miyako Ohkubo, and Mehdi Tibouchi. Unified, minimal and selectively randomizable structure-preserving signatures. In Yehuda Lindell, editor, *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, volume 8349 of *Lecture Notes in Computer Science*, pages 688–712. Springer, 2014.

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *J. Cryptology*, 21(2):149–177, 2008.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.

[BCC$^+$09] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Con-*

ference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings, volume 5677 of *Lecture Notes in Computer Science*, pages 108–125. Springer, 2009.

[BCG$^+$15] David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi. Sok: A comprehensive analysis of game-based ballot privacy definitions. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 499–516. IEEE Computer Society, 2015.

[BCKL08] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and noninteractive anonymous credentials. In Ran Canetti, editor, *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008.*, volume 4948 of *Lecture Notes in Computer Science*, pages 356–374. Springer, 2008.

[BCKL09] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. Compact e-cash and simulatable vrfs revisited. In Hovav Shacham and Brent Waters, editors, *Pairing-Based Cryptography - Pairing 2009, Third International Conference, Palo Alto, CA, USA, August 12-14, 2009, Proceedings*, volume 5671 of *Lecture Notes in Computer Science*, pages 114–131. Springer, 2009.

[BCP$^+$11] David Bernhard, Véronique Cortier, Olivier Pereira, Ben Smyth, and Bogdan Warinschi. Adapting Helios for provable ballot privacy. In Vijay Atluri and Claudia Díaz, editors, *Computer Security - ESORICS 2011 - 16th European Symposium on Research in Computer Security, Leuven, Belgium, September 12-14, 2011. Proceedings*, volume 6879 of *Lecture Notes in Computer Science*, pages 335–354. Springer, 2011.

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer, 1998.

[BEM15] Pedro Bibiloni, Alex Escala, and Paz Morillo. Vote validatability in mix-net-based evoting. In Rolf Haenni, Reto E. Koenig, and Douglas Wikström, editors, *E-Voting and Identity - 5th International Conference, VoteID 2015, Bern, Switzerland, September 2-4, 2015, Proceedings*, volume 9269 of *Lecture Notes in Computer Science*, pages 92–109. Springer, 2015.

[Ben06] Josh Benaloh. Simple verifiable elections. In Dan S. Wallach and Ronald L. Rivest, editors, *2006 USENIX/ACCURATE Electronic Voting Technology Workshop, EVT'06, Vancouver, BC, Canada, August 1, 2006*. USENIX Association, 2006.

[BFI$^+$10] Olivier Blazy, Georg Fuchsbauer, Malika Izabachène, Amandine Jambert, Hervé Sibert, and Damien Vergnaud. Batch Groth-Sahai. In Jianying Zhou and Moti Yung, editors, *Applied Cryptography and Network Security, 8th International Conference, ACNS 2010, Beijing, China, June 22-25, 2010. Proceedings*, volume 6123 of *Lecture Notes in Computer Science*, pages 218–235, 2010.

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

[BG12] Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 263–280. Springer, 2012.

[BHHO08] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision Diffie-Hellman. In David Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 108–125. Springer, 2008.

[BPW12a] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the Fiat-Shamir heuristic and applications to helios. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 626–643. Springer, 2012.

[BPW12b] David Bernhard, Olivier Pereira, and Bogdan Warinschi. On necessary and sufficient conditions for private ballot submission. *IACR Cryptology ePrint Archive*, 2012:236, 2012.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993.*, pages 62–73. ACM, 1993.

[BS99] Mihir Bellare and Amit Sahai. Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 519–536. Springer, 1999.

[BW06] Xavier Boyen and Brent Waters. Compact group signatures without random oracles. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 427–444. Springer, 2006.

[BW07] Xavier Boyen and Brent Waters. Full-domain subgroup hiding and constant-size group signatures. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography - PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography, Beijing, China, April 16-20, 2007, Proceedings*, volume 4450 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2007.

[CCas08] Jan Camenisch, Rafik Chaabouni, and abhi shelat. Efficient protocols for set membership and range proofs. In Josef Pieprzyk, editor, *Advances in Cryptology - ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7-11, 2008. Proceedings*, volume 5350 of *Lecture Notes in Computer Science*, pages 234–252. Springer, 2008.

[CCC$^+$09] David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, Alan T. Sherman, and Poorvi L. Vora. Scantegrity II: end-to-end verifiability by voters of optical scan elections through confirmation codes. *IEEE Trans. Information Forensics and Security*, 4(4):611–627, 2009.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 1994.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. *European Transactions on Telecommunications*, 8(5):481–490, 1997.

[Cha01a] D. Chaum. Physical and digital secret ballot systems, August 2 2001. WO Patent App. PCT/US2001/002,883.

[Cha01b] David Chaum. SureVote: Technical overview. In *Proceedings of the Workshop on Trustworthy Elections (WOTE '01)*, 2001.

[CHP12] Jan Camenisch, Susan Hohenberger, and Michael Østergaard Pedersen. Batch verification of short signatures. *J. Cryptology*, 25(4):723–747, 2012.

[CKLM12] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable proof systems and applications. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 281–300. Springer, 2012.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 494–503. ACM, 2002.

[CP92] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer, 1992.

[CRS05] David Chaum, Peter Y. A. Ryan, and Steve A. Schneider. A practical voter-verifiable election scheme. In Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors, *Computer Security - ESORICS 2005, 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005, Proceedings*, volume 3679 of *Lecture Notes in Computer Science*, pages 118–139. Springer, 2005.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 1998.

[Dam] Ivan Damgård. On sigma-protocols. http://www.cs.au.dk/~ivan/Sigma.pdf.

[Dan13] Quynh Dang. Changes in federal information processing standard (FIPS) 180-4, secure hash standard. *Cryptologia*, 37(1):69–73, 2013.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DHLW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 613–631. Springer, 2010.

[DJ01] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *Public Key Cryptography, 4th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001, Cheju Island, Korea, February 13-15, 2001, Proceedings*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136. Springer, 2001.

[Eck10] Peter Eckersley. Browser versions carry 10.5 bits of identifying information on average. https://www.eff.org/deeplinks/2010/01/tracking-by-user-agent, 2010.

[edn] eDNI. http://www.dnielectronico.es/PortalDNIe/. Accessed: 2015-07-11.

[EG14] Alex Escala and Jens Groth. Fine-tuning Groth-Sahai proofs. In Hugo Krawczyk, editor, *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, volume 8383 of *Lecture Notes in Computer Science*, pages 630–649. Springer, 2014.

[EGHM16] Alex Escala, Sandra Guasch, Javier Herranz, and Paz Morillo. Universal cast-as-intended verifiability. In Jeremy Clark, Sarah Meiklejohn, Peter Y. A. Ryan, Dan S. Wallach, Michael Brenner, and Kurt Rohloff, editors, *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*, volume 9604 of *Lecture Notes in Computer Science*, pages 233–250. Springer, 2016.

[EHK$^+$13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge L. Villar. An algebraic framework for diffie-hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *Advances* *in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 129–147. Springer, 2013.

[EHK$^+$17] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Luis Villar. An algebraic framework for diffie-hellman assumptions. *J. Cryptology*, 30(1):242–288, 2017.

[FLM11] Marc Fischlin, Benoît Libert, and Mark Manulis. Non-interactive and re-usable universally composable string commitments with adaptive security. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*, pages 468–485. Springer, 2011.

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM J. Comput.*, 29(1):1–28, 1999.

[FOO92] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In Jennifer Seberry and Yuliang Zheng, editors, *Advances in Cryptology - AUSCRYPT '92, Workshop on the Theory and Application of Cryptographic Techniques, Gold Coast, Queensland, Australia, December 13-16, 1992, Proceedings*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251. Springer, 1992.

[FPV09] Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Transferable constant-size fair e-cash. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *Cryptology and Network Security, 8th International Conference, CANS 2009, Kanazawa, Japan, December 12-14, 2009. Proceedings*, volume 5888 of *Lecture Notes in Computer Science*, pages 226–247. Springer, 2009.

[Fre10] David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 44–61. Springer, 2010.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[Fuc11] Georg Fuchsbauer. Commuting signatures and verifiable encryption. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*, pages 224–245. Springer, 2011.

[Gam84] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1984.

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

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GGP15] David Galindo, Sandra Guasch, and Jordi Puiggali. 2015 Neuchâtel's cast-as-intended verification mechanism. In Rolf Haenni, Reto E. Koenig, and Douglas Wikström, editors, *E-Voting and Identity - 5th International Conference, VoteID 2015, Bern, Switzerland, September 2-4, 2015, Proceedings*, volume 9269 of *Lecture Notes in Computer Science*, pages 3–18. Springer, 2015.

[GHV12] David Galindo, Javier Herranz, and Jorge L. Villar. Identity-based encryption with master key-dependent message security and leakage-resilience. In Sara Foresti, Moti Yung, and Fabio Martinelli, editors, *Computer Security - ESORICS 2012 - 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings*, volume 7459 of *Lecture Notes in Computer Science*, pages 627–642. Springer, 2012.

[Gjø10] Kristian Gjøsteen. Analysis of an internet voting protocol. *IACR Cryptology ePrint Archive*, 2010:380, 2010.

[Gjø11] Kristian Gjøsteen. The Norwegian internet voting protocol. In Aggelos Kiayias and Helger Lipmaa, editors, *E-Voting and Identity - Third International Conference, VoteID 2011, Tallinn, Estonia, September 28-30, 2011, Revised Selected Papers*, volume 7187 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2011.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In Robert Sedgewick, editor, *Proceedings of the 17th Annual*

ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA, pages 291–304. ACM, 1985.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[GNR$^+$01] Ed Gerck, C. Andrew Neff, Ronald L. Rivest, Aviel D. Rubin, and Moti Yung. The business of electronic voting. In Paul F. Syverson, editor, *Financial Cryptography, 5th International Conference, FC 2001, Grand Cayman, British West Indies, February 19-22, 2002, Proceedings*, volume 2339 of *Lecture Notes in Computer Science*, pages 234–259. Springer, 2001.

[GO14] Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. *J. Cryptology*, 27(3):506–543, 2014.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pages 97–111. Springer, 2006.

[GOS12] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11:1–11:35, 2012.

[GPS08] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[Gro04] Jens Groth. Evaluating security of voting schemes in the universal composability framework. In Markus Jakobsson, Moti Yung, and Jianying Zhou, editors, *Applied Cryptography and Network Security, Second International Conference, ACNS 2004, Yellow Mountain, China, June 8-11, 2004, Proceedings*, volume 3089 of *Lecture Notes in Computer Science*, pages 46–60. Springer, 2004.

[Gro06] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology - ASIACRYPT 2006, 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006, Proceedings*, volume 4284 of *Lecture Notes in Computer Science*, pages 444–459. Springer, 2006.

[Gro07] Jens Groth. Fully anonymous group signatures without random oracles. In Kaoru Kurosawa, editor, *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December*

2-6, 2007, Proceedings, volume 4833 of *Lecture Notes in Computer Science*, pages 164–180. Springer, 2007.

[GS12] Jens Groth and Amit Sahai. Efficient noninteractive proof systems for bilinear groups. *SIAM J. Comput.*, 41(5):1193–1232, 2012.

[GSW10] Essam Ghadafi, Nigel P. Smart, and Bogdan Warinschi. Groth-Sahai proofs revisited. In Phong Q. Nguyen and David Pointcheval, editors, *Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings*, volume 6056 of *Lecture Notes in Computer Science*, pages 177–192. Springer, 2010.

[GV10] Rojan Gharadaghy and Melanie Volkamer. Verifiability in electronic voting - explanations for non security experts. In Robert Krimmer and Rüdiger Grimm, editors, *Electronic Voting 2010, EVOTE 2010, 4th International Conference, Co-organized by Council of Europe, Gesellschaft für Informatik and E-Voting.CC, July 21st - 24th, 2010, in Castle Hofen, Bregenz, Austria*, volume 167 of *LNI*, pages 151–162. GI, 2010.

[HJ16] Dennis Hofheinz and Tibor Jager. Tightly secure signatures and public-key encryption. *Des. Codes Cryptography*, 80(1):29–61, 2016.

[HK07] Dennis Hofheinz and Eike Kiltz. Secure hybrid encryption from weakened key encapsulation. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, volume 4622 of *Lecture Notes in Computer Science*, pages 553–571. Springer, 2007.

[ISO06] Information technology - automatic identification and data capture techniques - qr code 2005 bar code symbology specification. ISO/IEC 18004:2000, 2006.

[JRF09] Rui Joaquim, Carlos Ribeiro, and Paulo Ferreira. Veryvote: A voter verifiable code voting system. In Peter Y. A. Ryan and Berry Schoenmakers, editors, *E-Voting and Identity, Second International Conference, VOTE-ID 2009, Luxembourg, September 7-8, 2009. Proceedings*, volume 5767 of *Lecture Notes in Computer Science*, pages 106–121. Springer, 2009.

[Kil90] Joe Kilian. *Uses of randomness in algorithms and protocols*. MIT Press, 1990.

[KP98] Joe Kilian and Erez Petrank. An efficient noninteractive zero-knowledge proof system for NP with general assumptions. *J. Cryptology*, 11(1):1–27, 1998.

[KV13] Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. *J. Cryptology*, 26(4):714–743, 2013.

[Lam79] L. Lamport. Constructing digital signatures from a one-way function. Technical report, October 1979.

[LOS$^+$10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 62–91. Springer, 2010.

[LY12] Benoît Libert and Moti Yung. Non-interactive cca-secure threshold cryptosystems with adaptive security: New framework and constructions. In Ronald Cramer, editor, *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, volume 7194 of *Lecture Notes in Computer Science*, pages 75–93. Springer, 2012.

[Mei09] Sarah Meiklejohn. An extension of the Groth-Sahai proof system. Master's thesis, 2009.

[Mil85] Victor S. Miller. Use of elliptic curves in cryptography. In Hugh C. Williams, editor, *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer, 1985.

[MMP02] Dahlia Malkhi, Ofer Margo, and Elan Pavlov. E-voting without 'cryptography'. In Matt Blaze, editor, *Financial Cryptography, 6th International Conference, FC 2002, Southampton, Bermuda, March 11-14, 2002, Revised Papers*, volume 2357 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2002.

[MRS88] Silvio Micali, Charles Rackoff, and Bob Sloan. The notion of security for probabilistic cryptosystems. *SIAM J. Comput.*, 17(2):412–426, 1988.

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

[MSF10] Sarah Meiklejohn, Hovav Shacham, and David Mandell Freeman. Limitations on transformations from composite-order to prime-order groups: The case of round-optimal blind signatures. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 519–538. Springer, 2010.

[NBV14] Stephan Neumann, Jurlind Budurushi, and Melanie Volkamer. *Analysis of Security and Cryptographic Approaches to Provide Secret and Verifiable Electronic Voting*, chapter 2, pages 27–61. Design, Development, and Use of Secure Electronic Voting Systems. IGI Global, 2014.

[NS12] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. *SIAM J. Comput.*, 41(4):772–814, 2012.

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

[OKNV12] Maina M. Olembo, Anna Kahlert, Stephan Neumann, and Melanie Volkamer. Partial verifiability in POLYAS for the GI elections. In Manuel J. Kripp, Melanie Volkamer, and Rüdiger Grimm, editors, *5th International Conference on Electronic Voting 2012, (EVOTE 2012), Co-organized by the Council of Europe, Gesellschaft für Informatik and E-Voting.CC, July 11-14, 2012, Castle Hofen, Bregenz, Austria*, volume 205 of *LNI*, pages 95–109. GI, 2012.

[Ore87] Yair Oren. On the cunning power of cheating verifiers: Some observations about zero knowledge proofs (extended abstract). In *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*, pages 462–471. IEEE Computer Society, 1987.

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 191–208. Springer, 2010.

[OT12] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure unbounded inner-product and attribute-based encryption. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 349–366. Springer, 2012.

[OT15] Tatsuaki Okamoto and Katsuyuki Takashima. Achieving short ciphertexts or short secret-keys for adaptively secure general inner-product encryption. *Des. Codes Cryptography*, 77(2-3):725–771, 2015.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference* *on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.

[RS91] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer, 1991.

[SC12] Jae Hong Seo and Jung Hee Cheon. Beyond the limitation of prime-order bilinear groups, and round optimal blind signatures. In Ronald Cramer, editor, *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, volume 7194 of *Lecture Notes in Computer Science*, pages 133–150. Springer, 2012.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.

[Seo12] Jae Hong Seo. On the (im)possibility of projecting property in prime-order setting. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 61–79. Springer, 2012.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[Sha07] Hovav Shacham. A Cramer-Shoup encryption scheme from the linear assumption and from progressively weaker linear variants. *IACR Cryptology ePrint Archive*, 2007:74, 2007.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997.

[SJ00] Claus-Peter Schnorr and Markus Jakobsson. Security of signed ElGamal encryption. In Tatsuaki Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings*, volume 1976 of *Lecture Notes in Computer Science*, pages 73–89. Springer, 2000.

[SK95] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme - A practical solution to the implementation of a voting booth. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology - EUROCRYPT '95, International Conference on the Theory and Application of Cryptographic Techniques, Saint-Malo, France, May 21-25, 1995, Proceeding*, volume 921 of *Lecture Notes in Computer Science*, pages 393–403. Springer, 1995.

[TW10] Björn Terelius and Douglas Wikström. Proofs of restricted shuffles. In Daniel J. Bernstein and Tanja Lange, editors, *Progress in Cryptology - AFRICACRYPT 2010, Third International Conference on Cryptology in Africa, Stellenbosch, South Africa, May 3-6, 2010. Proceedings*, volume 6055 of *Lecture Notes in Computer Science*, pages 100–113. Springer, 2010.

[TY98] Yiannis Tsiounis and Moti Yung. On the security of ElGamal based encryption. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography, First International Workshop on Practice and Theory in Public Key Cryptography, PKC '98, Pacifico Yokohama, Japan, February 5-6, 1998, Proceedings*, volume 1431 of *Lecture Notes in Computer Science*, pages 117–134. Springer, 1998.

[Vil12] Jorge Luis Villar. Optimal reductions of some decisional problems to the rank problem. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 80–97. Springer, 2012.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 160–164. IEEE Computer Society, 1982.