Zero Knowledge Protocols and Applications

Pavlos Ioannis Pyrros Chaidos

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Security & Crime Science

University College London

2017

I, Pavlos Ioannis Pyrros Chaidos, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

The historical goal of cryptography is to securely transmit or store a message in an insecure medium. In that era, before public key cryptography, we had two kinds of people: those who had the correct key, and those who did not. Nowadays however, we live in a complex world with equally complex goals and requirements: securely passing a note from Alice to Bob is not enough. We want Alice to use her smartphone to vote for Carol, without Bob the tallier, or anyone else learning her vote; we also want guarantees that Alice's ballot contains a single, valid vote and we want guarantees that Bob will tally the ballots properly. This is in fact made possible because of zero knowledge protocols.

This thesis presents research performed in the area of zero knowledge protocols across the following threads: we relax the assumptions necessary for the Damgård, Fazio and Nicolosi (DFN) transformation, a technique which enables one to collapse a number of three round protocols into a single message. This approach is motivated by showing how it could be used as part of a voting scheme. Then we move onto a protocol that lets us prove that a given computation (modeled as an arithmetic circuit) was performed correctly. It improves upon the state of the art in the area by significantly reducing the communication cost.

A second strand of research concerns multi-user signatures, which enable a signer to sign with respect to a set of users. We give new definitions for important primitives in the area as well as efficient instantiations using zero knowledge protocols.

Finally, we present two possible answers to the question posed by voting receipts. One is to maximise privacy by building a voting system that provides receipt-freeness automatically. The other is to use them to enable conventual and privacy preserving vote copying.

Acknowledgements

I owe a great debt of thanks to Jens Groth for his supervision, guidance and support in this adventure. I must also thank Allan Sikk for his grounded perspective and our productive talks. Thanks are also due to Yvo Desmedt for his trust and motivation. My coauthors, remote or local have all been huge positive influences in too many ways to mention.

I want to give a special thanks to the UCL InfoSec group, and to the folks at room 6.22 specifically. You all made this feel like a home away from home and I could not be more thankful. I know we'll meet again, but it still feels sad to leave.

I also want to single out my mother for her love and support throughout. I'd like to thank Nicolas for all the chats, pointless and meaningful, at every odd hour. Last but not least, Sundy for her love, support, and inspiration.

Thanks!

Pyrros

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The historical goal of cryptography is to securely transmit or store a message via an insecure medium. In that era, before public key cryptography, we had two kinds of people: those who had the correct key, and those who did not. Nowadays however, we live in a complex world with equally complex goals and requirements: securely passing a note from Alice to Bob is not enough. We want Alice to be able to use her smartphone to vote for Carol, without Bob the tallier, or anyone else learning her vote, we want guarantees that Alice's ballot contains a single, valid vote and we want guarantees that Bob will tally the ballots properly. This is in fact made possible because of zero knowledge protocols.

Generalizing from the voting example, in a zero knowledge protocol a prover wishes to convince a verifier that a particular statement is true. Alice, in our example, would want to convince Bob that her (encrypted) ballot only contains a single vote (as opposed to 5, or simply random data). Alice, acting as the prover has some private information (a witness) which would convince anyone, beyond the shadow of a doubt, that the statement is true. However, the prover does not wish to reveal this witness. Alice could easily convince Bob by showing him the contents of her ballot i.e. her vote, as well as any randomness used to encrypt it, but this would completely defeat the secrecy of her ballot. Zero knowledge protocols allow us to satisfy both parties: without revealing the witness to anyone, they make it infeasible (or even impossible) for the prover to convince the verifier of false statements.

Balancing security and privacy is hard enough on its own, but to fully capture our voting example we need a third pillar: efficiency. After all, Alice wants to vote on her smartphone, and Bob might need to process and tally thousands if not millions of ballots. Efficiency can manifest itself in different ways: processing time, storage space or interactivity, the need for back and forth communication between the two parties.

In this thesis we will study four different areas, each time improving upon the state of the art in one of the three pillars. We will consider: (a) Non-interactive zero knowledge designated verifier arguments, where we remove a computational assumption and replace it with one that can be fully realised in applications, (b) The evaluation of arithmetic circuits where we improve on existing work in terms of processing time as well as communication, (c) Accountable Ring Signatures and Group signatures, where we provide a compact accountable ring signature, give a new, more precise definition of group signatures and demonstrate how one can obtain a group signature scheme from accountable ring signatures. Finally, (d) we will examine the topic of receipt-freeness in voting from two opposite directions: first by building a voting system that achieves a strong version of the property, and second by studying a secure vote-copying protocol built on top of a voting system lacking in receipt-freeness.

In many cases the improvements listed in the above contributions are not free: often there is a price to be paid in terms of efficiency or stronger assumptions (or equivalently, potentially weaker security). We will discuss these issues and make comparisons with works tackling similar problems with different methods in the following chapters. In the rest of this chapter we will give a succinct introduction to each problem area in order to establish some context for our contributions.

1.1 Non-Interactive Zero Knowledge

The original notion of zero knowledge, due to Goldwasser, Micali, and Rackoff [GMR89], captures interactive protocols between a prover and a verifier. Such protocols enable the prover to convince the verifier that a given statement is true, whilst satisfying the following properties:

Completeness If the statement is indeed true, the protocol will convince the verifier.

Soundness If the statement is false, it will be infeasible for the prover to convince the verifier, even if she attempts to cheat by deviating from the protocol.

Zero Knowledge The protocol gives the verifier only a single bit of information: that the statement is true.

Such statements can also have witnesses associated with them. For example, Alice may state that her ballot contains a valid vote. A possible witness would be the vote and the randomness used to produce the ballot. Given a statement and a witness, verification is simple. The object of zero knowledge protocols is to allow verification to be performed accurately without revealing the witness, or in fact leaking any other information.

The concept of the witness is tightly coupled with the working definition of zero knowledge: that there exists a way to simulate the execution of the protocol regarding a true statement without needing to know the corresponding witness. Intuitively, a simulated execution cannot give information regarding the witness because the witness was never a part of it. However, if such simulated executions are indistinguishable from real ones, then real ones cannot leak information either.

When real and simulated transcripts follow identical distributions, we have perfect zero-knowledge. We can relax this to statistical zero knowledge, where the two distributions are statistically close, or even computational zero knowledge, where the distributions are indistinguishable only by computationally-bound adversaries.

Similarly, perfect soundness means that the prover cannot cheat, statistical soundness means that any cheating prover will fail to prove a false statement with overwhelming probability and computational soundness further relaxes this to only cover polynomially-bound provers. Computationally sound protocols are called arguments, as opposed to proofs.

The motivation behind such relaxations is efficiency: we know that perfect soundness precludes short (sub-linear) proofs for non-trivial languages [GVW02, GH98]. At the same time, succinct arguments were presented by Kilian [Kil92] in 4 rounds and later compressed to a single round by Micali [Mic94] via the Fiat-Shamir heuristic. Round complexity is a particularly important parameter, moreso when they only require a single move and are thus non-interactive. We call such Non-Interactive Zero Knowledge protocols NIZKs [BFM88].

The Fiat-Shamir transformation [FS87] is a widely used tool to transform a Sigma protocol into a NIZK proof. It is conceptually simple, and efficient to implement: we simply replace the verifier's challenge with a hash of the protocol's transcript thus far. The transformation is secure in the random oracle model (ROM) [BR93] where the hash function is idealised into a truly random function. Furthermore, the model also allows for the oracle to be programmed (even adaptively) if required.

Unfortunately, the random oracle model is too strong. Canetti, Goldreich and Halevi [CGH04] proved that for any candidate hash function, one can build encryption schemes and signature schemes which are secure under the ROM, but trivially insecure in practice. This was later generalised by Goldwasser and Kalai [GK03] who produced signature schemes that would fail for any instantiation while being provably secure in the ROM. Nevertheless, it is still widely used as there is evidence that non-interactive zero knowledge with sub-linear communication cost requires unfalsifiable assumptions [GW11] i.e. assumptions that cannot be modeled as a security game whose outcome can be efficiently checked.

One alternative to the Fiat-Shamir transformation is the one by Damgård, Fazio and Nicolosi (DFN) [DFN06], which can be used to transform a restricted class of Sigma-protocols into NIZK arguments for a designated verifier. In their transformation, instead of using a hash to replace the verifier's challenge, the verifier registers an encrypted challenge ahead of time. Because the cryptosystem used for this is additively homomorphic, and the prover's last message is linear in the challenge, the prover is able to complete the protocol by producing a ciphertext containing her answer. However, the security proof for their construction requires a complexity leveraging assumption: i.e. a class of (possibly super-polynomial) adversaries strong enough to completely break one computational problem but at the same time weak enough that a second problem remains hard. In contrast with gap assumptions (where two problems are separated into "easy" and "hard"), both problems are considered hard for polynomial adversaries. The additional tiering into "hard" and "very hard" only becomes relevant inside certain security reductions.

In chapter 4 we will revisit the DFN transformation with the goal of removing the complexity leveraging assumption they require. To do that, we use the concept of culpable soundness [GOS12], which is a relaxation of standard soundness in that a successful adversary needs to not only prove a false statement, but also provide a proof of this falsehood, called a guilt witness, in a fashion established ahead of time. The nature of guilt witnesses can be different for every application. In the case of the DFN transformation, they take the form of decryption keys, allowing us to peek inside ciphertexts that would be brute forced under the original complexity leveraging assumption. It might seem that we are essentially formalizing the same concept in a different way, but that is false. The second part of our contribution is to demonstrate a practical application of NIZKs (namely, a simplified voting system) where the guilt witness is available in the environment. Thus, whereas the security of the original construction is conditioned on the complexity leveraging assumption for all applications, we manage to show that for certain applications the equivalent assumption (i.e. the infeasibility of a proof for a false statement absent a guilt witness) is always satisfied.

This is done by building a minimal voting system: the decryption key for the ballots serves as the guilt witness. Since the key exists in the environment, and soundness does not rely on it being secret, we are able to make any successful adversary culpable by simply giving them the key.

1.2 Protocols for Arithmetic Circuits

Arithmetic circuits are a more expressive variant of binary circuits: instead of restricting the values carried by wires to 1s and 0s, we consider values from a finite field $\mathbb{Z}_p$. We mainly consider addition and multiplication gates (note that AND gates are essentially multiplication gates and OR gates can be built out of addition and multiplication –with the help of a constant "-1" wire). Apart from their generality, arithmetic circuits are also practical: there exists a compiler of restricted C programs to arithmetic circuits.

This practicality of arithmetic circuits is also manifest in their use in the verifiable computation setting. Verifiable computation (VC) setting eschews zero knowledge in favour of efficiency. By using succinct non-interactive arguments of knowledge (SNARKs) [GGPR13] based on bilinear maps, Pinocchio [PHGR13] manages to produce a verifiable computation system with constant-size proofs where verification times for a computation are faster than running the computation locally. Most of this efficiency is preserved when adding zero knowledge to our requirements.

When we consider protocols based on the discrete log problem, the most efficient ones are those of Groth [Gro09b] and Seo [Seo11], both achieving square root communication. ZKBoo [GOM16] relies only on collision resistant hash functions and achieves excellent performance but produces large proofs (linear in both the circuit size and the desired soundness parameter).

In chapter 5 we will give a discrete log protocol for the evaluation of arithmetic circuits that manages to improve upon the state of the art in the following ways:

- We can achieve logarithmic proof size by using a logarithmic number of rounds.
- For constant rounds, we can match the 5 move interaction cost of Seo's protocol, while also achieving lower communication and computation cost than Groth's 7 move protocol, improving upon what would be the "best of both worlds".
- We are able to tune the number of rounds to any intermediate point between the logarithmic and the constant version, enabling us to balance communication and compu-

tation cost as required.

- The protocol comes with a prototype python implementation, which manages to outperform Pinocchio's prover (but not the verifier).

1.3 Multiuser Signatures

Group and Ring signatures allow a team member to sign with respect to that team without disclosing her identity. A trivial solution would be to create a single signing key for the team and give it to every member. Obviously, this solution is extremely poor: there is no accountability, no ability to remove users (other than starting over) and members of multiple teams need to be given keys for every team. Group signatures [CvH91] offer solutions to these problems: they provide a tracing algorithm that can be run in case of abuse, so as to reveal the individual who produced the signature on a malicious message. The group is overseen by a manager who is also responsible for adding users (dynamic groups) or adding and removing users (fully dynamic groups) from the group.

Ring signatures [RST01] do away with the manager and effectively let users manage themselves: any user is able to define a ring (i.e. a set of users that includes herself) and sign with respect to it. Every signature can be attached to a different ring. However, ring signatures typically lack a tracing functionality. Accountable ring signatures are a more recent concept that attempts to combine the best features of groups and rings: users are still free to define the ring they are signing against, but accountability is still enforced. Furthermore, ring signature schemes can allow signers to specify who will be the tracing authority. This can enable interoperability by allowing the same infrastructure and keys to be reused across different settings by only changing the authority specified. For example, Alice would use the same credentials to sign as a member of her employer's information security team (with the team members as the ring, and their manager as the tracing authority) as well as a member of her local science fiction book club (with the club members forming the ring, and Bob, the club president as the authority).

Existing definitions for dynamic group signatures are oftentimes built around a specific instantiation, and in that way inherit some of its flaws. As we will see, this is in fact the case with some constructions based around revocation lists. Revocation lists are one method used to achieve revocation i.e. the removal of a member from a group. They require that the signer also provides proof that her identity is not featured in a list of revoked users that is published by the manager. Intuitively, this might seem to capture our security requirement: revoking a user prevents them from signing afterwards. However, this proves to be imprecise: what one would reasonably expect is that users are able to sign with respect to time periods following them joining the group and (if applicable) preceding their revocation. Schemes such as [NFHF09,LPY12b,LPY12a], which follow the revocation list paradigm fail to consider a simple back-dating attack: a user who joined the group in June is not prevented from signing with regards to the group membership as it was in March. In the absence of a reliable opening oracle, this may implicitly frame past members. In this thesis we will study both group and ring signatures. We will first address accountable ring signatures (Chap. 6), starting with definitions before moving on to a construction in the discrete logarithm setting. Our construction is based on the ring signature of Groth and Kohlweiss [GK14] but adds accountability as well as some performance enhancements (namely n-ary trees instead of binary, and substituting Pedersen commitments for ElGamal ciphertexts when possible).

We will then move to group signatures (Chap. 7) and give a rigorous set of definitions for their security. We also give a full description of the backdating attack, and show how affected schemes may be repaired. Finally, we show how existing models for group signatures can be described in our framework. Finally, we link the two concepts, and show how group signatures and can be built from accountable ring signatures, with our scheme used as a concrete example.

1.4 Voting and Receipt Freeness

Electronic Voting has been a prominent cryptographic application for the last 30 years, after Chaum's early work [Cha81]. However, Internet voting for general elections is only regularly practiced in a limited number of countries [SFD$^+$14]. A few more use electronic in-person voting (with cryptography being deployed in varying degrees). By contrast, financial transactions over the internet are near-ubiquitous. If Alice can buy a new phone from her computer (or a new computer from her phone) why is internet voting so hard?

First off, the privacy requirements are much stricter. If Alice orders a pizza over the internet she might reasonably expect her choice of toppings or sides to be private; using more recent payment systems, the identity of the business she's paying might be obscured as well. The amount transacted however needs to be made known to her bank, which Alice must (at least partly) trust. Part of this trust may also be mitigated via regulation and state oversight.

When voting however, Alice expects a higher degree of privacy. Ideally, the only information that's disclosed is whether she voted or not (if that). In practice, some voting systems might reveal her ballot, but only after anonymizing it, and usually in a restricted setting. To achieve this via cryptography we use systems based on either blind signatures [Cha83] or homomorphic encryption such as [Elg85]. Blind signatures enable a voter to anonymously demonstrate her eligibility, reducing the problem to that of anonymously submitting the signed vote. Systems where votes encrypted via a homomorphic scheme allow for two solutions: they may be passed through a series of shuffling servers (a Mixnet), or they may be tallied homomorphically, where an encryption of the tally is produced directly by summing the encrypted votes. Zero knowledge protocols are used to ensure correct behaviour in all cases.

Even so, voting introduces an extra hurdle: we might need to protect Alice from herself. In particular we should not leave Alice in a position where she is able to prove how she voted, as that would leave her open to coercion. This property, introduced by Benaloh and Tuinstra [BT94], is receipt freeness, i.e. the requirement that voters are unable (even if willing) to construct receipts for their ballots. A receipt in this context allows its holder to check whether a ballot is in support of a particular candidate (e.g. by providing the randomness used in constructing the ballot). Unfortunately, preventing Alice from having a receipt rules out most of the easy solutions to verifiability [CMFPT06] necessitating additional assumptions such as trusted hardware, multi-party protocols or untappable channels.

In this thesis we will study two modifications of the Helios voting system. Helios, is a well-studied and widely deployed internet voting system based on homomorphic tallying. It is proposed as a system for elections in low coercion environments. Because it runs as a javascript application, users in control of their web browser are able to produce receipts by running simple javascript commands (see Sect. 8.2.1).

In chapter 8 we investigate the lack of receipt freeness as an opportunity for controlled ballot copying. Ballot copying without controls was pointed out in [CS11, CS13] and later developed into controlled ballot copying by [DC12]. Desmedt and Chaidos gave a vote-copying protocol utilising the divertibility of the proofs used in Helios. However, in order to maintain compatibility with Helios their protocol was only zero knowledge against honest verifiers (or copiers). In addition their privacy definition was informal, which as pointed out in [SB13] fails to highlight the privacy implications of the protocol itself versus those implied by the act of allowing copies. The work in this thesis repairs both issues: first we give a formal definition of privacy for ballot copying, and second, we extend the original Ballot copying protocol so as to achieve full zero knowledge.

In a different direction, chapter 9 will address the lack of receipt freeness by introducing BeleniosRF: a Helios inspired voting system which uses the encryption scheme of [BFPV11a] to achieve a strong variant of receipt freeness. Signed Randomizable Ciphertexts enable an intermediate server to rerandomise ballots before posting but prevents tampering or wholesale replacement. This ensures that a voter is unable to produce a receipt even if she were to disclose all of her information. This is similar to the notion of RCCA [CKN03] encryption, which combines the strong tamper-resistance of CCA-secure encryption with a rerandomisation functionality. However, since it does not involve signatures, RCCA security cannot guard against wholesale replacement, but only against tampering.

1.5 Structure

The next chapter 2 covers related work, and highlights the motivations and techniques behind the results in this thesis. Chapter 3 covers background concepts and gives definitions that will be used throughout the text. In some cases, highly domain-specific concepts will be deferred until the chapter in which they will be relevant.

The rest of the chapters are based around different projects. They may be read independently for the most part with any exceptions being mentioned in the text. Thematically, they form three loose groups. In the first group, chapter 4 covers the revisiting of the DFN transformation which, while being applicable to only a small subset of protocol, can still be used in a wide range of applications. After that, chapter 5 focuses exclusively on calculations that can be represented as arithmetic circuits. This produces a highly efficient family of protocols under without relying on any strong assumptions (for the online version).

The next two chapters, 6 and 7 cover multiuser signatures. We start by giving a definition and a generic construction for accountable ring signatures along with a very size-efficient instantiation. We also demonstrate how accountable ring signatures (ARS) can be used to derive both ring signatures and group signatures. Next, in chapter 7 we focus on giving new definitions for fully dynamic group signatures. Our interest is motivated both by some weaknesses present in existing definitions, as well as the presence of our ARS based construction.

Finally the last two chapters consider two divergent approaches to the issue of non receipt-free voting systems, such as Helios. In chapter 8 we exploit the presence of receipts to build a secure system allowing for consentual ballot copying. Opposing this, in chapter 9 we present BeleniosRF, a derivative of Helios designed to achieve receipt freeness without requiring any extra work from the voter.

Chapter 10 concludes this work and offers some potential future directions.

Publications.

- The research presented in chapter 4 was joint work with Jens Groth and published in PKC 2015 as [CG15].
- The research presented in chapter 5 was joint work with Jonathan Bootle, Andrea Cerulli, Jens Groth and Christophe Petit, and published in EUROCRYPT 2016 as [BCC+16b].
- The research presented in chapter 6 was joint work with Jonathan Bootle, Andrea Cerulli, Essam Ghadafi, Jens Groth and Christophe Petit, and was published in ESORICS 2015 as [BCC+15].
- The research presented in chapter 7 was joint work with Jonathan Bootle, Andrea Cerulli, Essam Ghadafi and Jens Groth, and was published in ACNS 2016 as [BCC+16a].
- The research presented in chapter 8 is joint work with Yvo Desmedt, and significantly updates work that was published in ESORICS 2012 as [DC12].
- The research presented in chapter 9 is joint work with Véronique Cortier, Georg Fuchsbauer and David Galindo, and was published in CCS 2016 as [CCFG16].

Chapter 2

Literature Review

This chapter covers relevant parts of existing work with the aim of establishing both context and motivation for the results that follow. First, we cover the area of zero-knowledge protocols, with a focus on non-interactivity as well as circuit-based protocols. Non-interactivity is a crucial feature for many applications as it is not feasible to assume that all participants in a system are online at the same time. At the same time, efficient circuit-based approaches can find wide usage in applications for which a tailor-made protocol is unavailable or infeasible. Multi-user signatures follow, where users are able to sign not as individuals but as member of a large set. Oftentimes, they use zero knowledge techniques to enforce correct behaviour amongst the (sometimes untrusted participants). Finally, we cover voting applications, focusing on solutions using homomorphic tallying such as Helios [Adi08]. Again, non-interactive zero knowledge protocols are a core component as malicious ballots need to be detected before being tallied into the result.

Parts of this chapter are derived from [CG15, BCC$^+$16b, BCC$^+$15, BCC$^+$16a, DC12, CCFG16] as well as the author's upgrade report.

2.1 Zero Knowledge

Zero-knowledge proofs were invented by Goldwasser et al. [GMR89]. Since then, they have been used in a number of cryptographic applications, particularly in areas covered in this text.

2.1.1 Non-Interactive Zero Knowledge

Σ-protocols are particular types of 3-move honest verifier zero-knowledge proofs that can be highly efficient. However, in many applications (e.g. in voting or in multiparty signatures) it is preferable for a protocol to be non-interactive [BFM88] with the prover preparing a proof with no need for direct input from the verifier. The Fiat-Shamir transformation [FS87] produces a non-interactive version of a Σ-protocol by substituting the verifier's challenge with the output of a hash function on the prover's statement and messages. The transformation can be proven secure in the random oracle model [BR93]. However, the random oracle model is regarded with some skepticism since there exist pathological protocols that can be proven secure in the random oracle model but fail in any real-world instantiation [CGH04, GK03].

The introduction of pairing-based techniques [GOS12, Gro06, GS12] has led to practically efficient non-interactive zero-knowledge proofs that can be used in the context of pairing-based cryptography. Recently pairing-based succinct non-interactive zero-knowledge arguments [Gro10b, GGPR13, PHGR13] have become very compact even for large scale statements, however, they rely on knowledge extractor assumptions over bilinear groups.

The above research yields non-interactive zero-knowledge proofs that are publicly verifiable. However, there are many settings where it suffices to have non-interactive zero-knowledge arguments intended for a designated verifier. Cramer and Shoup used universal hash proofs to build a highly efficient public-key encryption scheme [CS98, CS02], secure against chosen ciphertext attacks. Non-interactive proofs for a designated verifier for all languages in NP can be found in [BCNP04] in the key registration model where parties register keys.

Damgård, Fazio and Nicolosi (DFN) [DFN06] introduced an alternative to the Fiat-Shamir transformation. The DFN transformation works in the Registered Key Model (RKM) [BCNP04] where a verifier registers a public key and transforms a Σ-protocol with linear answer into a non-interactive zero-knowledge argument that can be verified by this specific verifier [JSI96]. The transformation works by having the verifier encrypt his challenge under an additively homomorphic encryption scheme and relies on the Σ-protocol having an answer that can be computed using linear algebra and the homomorphic property of the encryption scheme to enable the prover to complete an encrypted version of the answer in the Σ-protocol. Their construction is secure for a logarithmic number of proofs but soundness rests on a complexity leveraging assumption. A complexity leveraging assumptions states that there is a hardness separation between two hard problems, i.e. that there exist solvers for the first problem, running in time T, but no T-bound adversary can solve the second problem with more than negligible probability. This allows security reductions to raise a contradiction by solving the second, harder, problem using the adversary, and a solver for the first problem.

Ventre and Visconti [VV09] give an alternative proof of soundness for a construction based on a two ciphertext variation of the DFN transformation in the style of Naor and Yung [NY90]. They replace the complexity leveraging assumption by introducing a modification of culpable soundness[1] [GOS12] that they call *weak culpable soundness*. Culpable soundness restricts adversaries to being "aware" of the falsehood of the statement they are proving. *Weak* culpable soundness furthermore requires that the adversary is also aware of the fact that she has succeeded in producing a convincing proof of a false statement, by producing a second auxiliary proof to that effect.

In the DFN setting using weak culpable soundness would require the adversary to prove statements containing ciphertexts addressed to the designated verifier. It would be challenging to provide such an adversary with enough power to perform the required proofs without having knowledge of the verifier's secret decryption key (if the adversary knows the verifiers key, the system serves no function). To overcome this, in chapter 5 we will instead opt to construct the underlying protocol with the property that forged proofs reveal the challenge. This is enough to contradict the semantic security of the encryption scheme used for the designated verifier proof if a false proof is ever produced.

2.1.2 Circuit-Based Protocols

Since the introduction of non-interactive zero-knowledge proofs by Blum, Feldman and Micali [BFM88] much effort has been spent on reducing their size [Dam92, KP98, Gro10a, GGI$^+$14].

[1] Culpable soundness was also called co-soundness in an earlier version of [GOS12].

Both [Dam92] and [KP98] suggest protocols on circuit satisfiability, since SAT is NP-Complete and computer operations are naturally represented as circuits. Both protocols essentially operate in the hidden bits model, where a shared random string is “interpreted” in a way that hides information from the verifier while still preventing the prover from cheating. [Dam92] uses a shared sting of cubic size in the circuit size to encode the gates of the circuit as quadratic residues of large numbers (where squares encode 0 and non-squares 1). The prover then proves consistency by demonstrating that hidden values which are claimed to be equal multiply up to a square. However, the size of the proofs is as large as that of the common string. In [KP98] the dependency on the circuit size is reduced to $n \log n$ by using a different encoding with less redundancy.

This is further reduced to a quasi-linear dependency in the circuit in [Gro10a] by using probabilistically checkable proofs (PCPs) i.e. large proofs that may be convincingly checked by accessing only a small portion thereof. This helps compact the proof as the probability of a malicious prover failing is high in the base case, eliminating the need to boost it via repetition.

Gentry et al. [GGI$^+$14] used fully homomorphic encryption to construct zero-knowledge proofs where the communication complexity corresponds to the size of the witness. However, proofs cannot in general have communication that is smaller than the witness size unless surprising results about the complexity of solving SAT instances hold [GH98, GVW02].

It is useful to distinguish between zero-knowledge *proofs*, with statistical soundness, and zero-knowledge *arguments* with computational soundness. In general proofs can only have computational zero-knowledge, while arguments may have perfect zero-knowledge. Goldreich et al. [GMW91] showed that all languages in NP have zero-knowledge proofs while Brassard et al. [BCC88] showed that all languages in NP have zero-knowledge arguments with perfect zero-knowledge.

Kilian [Kil92] showed that in contrast to zero-knowledge proofs, zero-knowledge arguments can have very low communication complexity. His construction relied on the PCP theorem though, and did not yield a practical scheme.

Schnorr [Sch91] and Guillou and Quisquater [GQ88] gave early examples of practical zero-knowledge arguments for concrete number theoretic problems. Extending Schnorr's protocols, there have been many constructions of zero-knowledge arguments based on the discrete logarithm assumption. Cramer and Damgård [CD98] gave a zero-knowledge argument for arithmetic circuit satisfiability, which has linear communication complexity.

Currently the most efficient discrete logarithm based zero-knowledge arguments for arithmetic circuits are the ones by Groth [Gro09b] and Seo [Seo11], which are constant move arguments with a communication proportional to the square root of the circuit size.

A square root barrier. The works of [Gro09b, Seo11] share the common pattern of representing the circuit state as a matrix, using a commitment to represent each row, doing some protocol specific work and finishing with de-commiting a function of the committed rows. This seems to imply a square root lower bound on the size of such a proof: committing has a cost equal to the number of rows and revealing has a cost equal to the number of columns. Since both parts seem critical to proving soundness it seems that further improvements would either require a complete rethink of the representation and proof structure, or an intermediate "compression" step allowing us to reveal less data.

The work of [Gro09a] can be thought of as one such step, moving from a matrix to a cube to give a zero-knowledge argument with a cubic root communication complexity, but uses pairing-based cryptography instead of just relying on the discrete logarithm assumption. In the discrete log setting, [BG12] use an iterable reduction to reduce computation by compressing one problem instance into a smaller one (obviously, this hinges on the cost difference of the two instances being larger than that of the compression).

There are recent works giving a logarithmic communication complexity for specific languages. Bayer and Groth [BG13] show that one can prove that a polynomial evaluated at a secret committed value gives a certain output with a logarithmic communication complexity and Groth and Kohlweiss [GK14] show that one can prove that one out of N commitments contain 0 with logarithmic communication complexity. These results are for very specific types of statements (with low circuit depth) and the techniques do not seem to generalise to arbitrary NP languages.

An exciting line of research [Gro10b, Lip12, BCCT12, GGPR13, BCCT13, PHGR13, BCG$^+$13, BCTV14, GK14] has developed many proposals for succinct non-interactive arguments (SNARGs) yielding pairing-based constructions where the arguments consist of a constant number of group elements. However, they all rely on a common reference string (with a special structure) and non-falsifiable knowledge extractor assumptions. In contrast, the arguments we develop in Chapter 5 are based solely on the discrete logarithm assumption, and use a small common reference string which is independent of the circuit.

2.2 Multi-user Signatures

2.2.1 Group Signatures

After their introduction by [CvH91], a long line of research on group signatures has emerged. In the early years, security of group signatures was not well understood and early constructions were proven secure via informal arguments using various interpretations of their requirements.

Bellare et al. [BMW03] formalised the security definitions for static groups. In their model, the group manager (who also acts as the tracing authority) needs to be fully trusted. Later on, Bellare et al. [BSZ05] and Kiayias and Yung [KY06] provided formal security definitions for the more practical partially dynamic case. Also, [BSZ05] separated the tracing role from the group management. In both [BSZ05, KY06] models, members cannot leave the group once they have joined. More recently, Sakai et al. [SSE$^+$12] strengthened the security definitions for partially dynamic groups by defining *opening soundness*, ensuring that a valid signature only traces to one user.

The first practical and provably secure group signature was due to Ateniese et al. [ACJT00]. It yields signatures of constant size and are based on the DDH and the strong RSA assumptions, in the random oracle model. Their scheme was later improved by Camenisch and Lysyanskaya to allow efficient revocation of group member using dynamic accumulators [CL02]. Boneh et al. [BBS04] also constructed constant size group signatures under the strong Diffie-Hellman and the Decision Linear assumption in pairing groups. Other pairing-based schemes include [ACHdM05, NS04, CG05, BW07, Gro07, CKS09, LPY12a]. Recently, Langlois et al. [LLNW14] gave an efficient lattice-based group signature scheme supporting revocation, based on the hardness of approximating the shortest independent vectors problem in lattice of dimension n within a factor $\tilde{O}(n^{1.5})$.

Group Signatures Without Revocation. Constructions of group signatures in the random oracle model [BR93] include [CS97, CM98, ACJT00, BBS04, CL04, CG05, NS04, FI05, FY04, KY05, DP06, BCN$^+$10]. Constructions not relying on random oracles include [ACHdM05, Gro06, BW06, Gro07, BW07, AHO10].

Group Signatures With Revocation. Since revocation is an essential feature of group signatures, many researchers investigated the different approaches via which such a feature can be realised. One approach is for the group manager to change the group public key when members are removed and issue new group signing keys to all remaining legitimate members or allow them to update their old signing keys accordingly. This is the approach adopted by e.g. [TX03, CL02].

Bresson and Stern [BS01] realise revocation by requiring that the signer proves at the time of signing that her group membership certificate is not among those contained in a public revocation list. Another approach, which was adopted by e.g. [CL02, TX03, DKNS04, Ngu05], uses accumulators, i.e. functions that map a set of values into a fixed-length string and permit efficient proofs of membership.

Boneh, Boyen and Shacham [BBS04] showed that their static group signature scheme supports revocation since it allows members to update their signing keys according to the changes in the group without the involvement of the manager. Camenisch and Groth [CG05] also gave a construction that supports revocation. Song [Son01] gave a fully dynamic group signature with forward security.

A different approach for revocation known as *Verifier Local Revocation* (VLR), which needs relaxation of some of the security requirements, considered by Brickell [Bri04], was subsequently formalised by Boyen and Shacham [BS04] and further used in e.g. [NF05, LV09, LLNW14]. In VLR, the revocation information (i.e. revocation lists) is only sent to the verifiers (as opposed to both verifiers and signers) who can check whether a particular signature was generated by a revoked member. A similar approach is also used in Direct Anonymous Attestation (DAA) protocols [BCC04]. *Traceable Signatures* [KTY04] extend this idea, as the group manager can release a trapdoor for each member, enabling their signatures to be traced back to the individual user.

More recently, Libert, Peters and Yung [LPY12b, LPY12a] gave a number of efficient constructions of group signatures supporting revocation without requiring random oracles by utilizing the subset cover framework [NNL01] that was originally used in the context of broadcast encryption. [NFHF09] is another certificate-based scheme and achieves constant size as well as signing and verification time in the random oracle model. However, the security models of [LPY12b, LPY12a, NFHF09] all fail to account for users producing “backdated” signatures (i.e. signatures w.r.t to a time before them joining the group). While one would expect such signatures to be invalid, the models do not cover it and their constructions allow them to verify. As such, this may be used to (implicitly) implicate past users for the actions of current ones. In Chapter 7 we examine this attack in detail, fix the affected schemes and provide a strict definition for fully dynamic group signatures.

2.2.2 Ring Signatures

Ring Signatures were first introduced in [RST01], offering a way for users to sign on behalf of a set without the requirement for managers or registration. Their construction produces signatures with size linear in the group. A formal security model for ring signatures was provided by Bender et al. [BKM09].

Constant-size ring signatures can be based on the strong RSA assumption [DKNS04] and on pairing assumptions [Ngu05]. Very recently, Groth and Kohlweiss provided a ring signature scheme based on the discrete logarithm assumption in the random oracle model, which is asymptotically more efficient than previous ones.

2.2.3 Accountable Ring Signatures

Accountable ring signatures were informally defined by Xu and Yung [XY04]. Their security model mitigates the trust on the opener by using several openers and a threshold decryption mechanism. However, their construction requires a cumbersome registration protocol between the user, her smart card and the opener, whereas in our model a user needs only to be aware of the public key of an opener (the opener does not need to be aware of the user). This makes it simpler and more efficient to compartmentalize trust by using different openers for different contexts (e.g posting on a technical message board vs a political one) as there is no registration process to be duplicated.

The [DKNS04] construction, supports identity escrow in a model similar to that of accountable ring signatures, but without requiring openers to prove the opening is correct when they claim a particular user made a signature.

Traceable ring signatures [FS08] and linkable ring signatures [LWW04] also offer some restricted form of accountability. In traceable ring signatures, any couple of signatures produced by the same user and under the same tag will reveal her identity if they sign different messages. Two signatures by the same user for the same message and under the same tag will be distinguishable as duplicates, but without revealing the signer. In linkable group signatures, it is possible to efficiently decide whether two signatures were produced by the same user but without revealing his identity. Unique ring signatures [FZ13] encompass elements of both traceable and linkable ring signatures: signatures by the same user for the same message will be linked, but signatures of different messages have no special property.

2.3 Internet Voting

The technical feasibility for internet voting was first pointed out by Chaum [Cha81], and was followed by a large output of research, broadly split into three "schools" of ensuring anonymity. Mix-Based approaches, (also introduced in [Cha81], along with their voting application) operate by shuffling ballots before decrypting them so that the link between voter and vote is obscured. Solutions based on blind signatures [Cha83] focus on anonymous "endorsement" (i.e. signing) of ballots by authorities so that only registered voters can get their ballots endorsed while the authorities are unable to determine the voter behind a final ballot. Finally, solutions based on homomorphic tallying require a homomorphic encryption scheme so that ballots can be added together before they are decrypted, so that only an encryption of the tally requires decryption.

2.3.1 Homomorphic Tallying

Initial efforts in homomorphic tallying systems are due to Benaloh (previously Cohen) [CF85, BY86]. In such a system valid encrypted ballots can be added together in a straightforward manner to produce an encrypted sum of the votes. Both systems used a form of cut and choose approach where the validity of an encrypted ballot would be guaranteed by interactively revealing part of it in a random way they differ in that the later system supports multiple authorities to better manage trust. The randomness for the ballot proof was facilitated by means of a beacon. Later systems such as [CFSY96, CGS97a] use the Fiat-Shamir heuristic in place of a beacon in order to provide random challenges for the proofs of ballot validity. Both systems support multiple authorities but [CGS97a] is more efficient by using threshold encryption [DF90] rather than the more expensive technique of verified secret sharing [Fel87]. The system of Hirt & Sako [HS00] modify [CGS97a] to provide the newer security property of receipt freeness [BT94]. This is accomplished by having the authority prepare a list of encryptions of all possible options in a scrambled order and giving the voter a designated verifier proof of the correct ordering. Because of the designated verifier property (i.e the verifier is in the unique position of being able to forge proofs), the proof is of no value to an adversary. Shorty after the invention of the Paillier cryptosystem [Pai99] it was put to use in homomorphic tallying systems [DJN10, BFP$^+$01] as being additively homomorphic is preferable to the multiplicative homomorphic property of ElGamal. The system by Baudron et al. also supports a regional hierarchy that mirrors paper-based elections and also mentions the possibility of using randomisers to obtain receipt freeness. A later system by Damgård et al. [DGS03] improves on [CGS97a] by using a single ciphertext to encrypt multiple choices thus reducing the number of proofs required.

Helios [Adi08] (since the 2nd version) is an implemented voting platform with a design very similar to [CGS97a]. Since its introduction it has been attacked [ED10], repaired [CS13], modified to add functionality [DC12] and improved to provide everlasting privacy [DVDGA12]. It has been used in several real-world elections such as that of the president of the University of Louvain-la-Neuve in Belgium, and of the Board of Directors of the IACR since 2011 [IAC].

Kulyk *et al.* [KTV15] propose an extension of Helios where voters may later cancel their vote, still being indistinguishable from null votes submitted by the crowd. In addition to being difficult to deploy in practice, this scheme strongly relies on revoting. Hirt's scheme [Hir10] heavily depends on the existence of untappable channels. Selene [RRI15] proposes an enhancement for receipt-free schemes, to ease the verification step made by voters through tracking numbers.

A different approach is that of self-tallying systems such as [KY02, DJ03, Gro04a, HK11]. In such a system, a (somewhat expensive) pre-processing step ensures that the submitted encrypted ballots will sum to a decrypted or easily decryptable value. This requires that participants do not abstain since their vote is needed to facilitate decryption. In settings with a small number of participants this may be easily rectified by restarting the protocol, but for large groups it becomes necessary to re-introduce an election authority albeit with a very limited role. The presence of an authority also prevents the situation where the last person to vote holds the advantage of being able to know the result before everyone else.

2.3.2 Helios Ballot Copying

Due to the popularity and prominence of Helios, a number of works have examined its internals resulting in both attacks as well as additional, unplanned functionality.

Cortier & Smyth [CS11] attack a voter's privacy by means of a replay attack. In the base version of their attack, a ballot is recast either verbatim or with minor differences in the representation of the signatures by a number of parties under the control of the attackers. The checks performed by the then current Hellios ballot casting server were somewhat lax. In some scenarios the additional votes for the original voter's chosen party or candidate will significantly bias the election result, thus violating privacy. The authors offer the French legislative elections as an example of such a scenario. A more complex version of their attack involves a permutation of the voter's choices making the malicious ballots slightly harder to detect.

Esteghari & Desmedt [ED10] describe an attack which essentially installs a rootkit in the user's web-browser by exploiting a vulnerability in Adobe Reader. The rootkit then secretly changes the user's vote to a different one, and also hides any evidence of foul play. Helios, operating under the assumption that the user's browser is trustworthy, accepts the changed vote instead of the intended one.

The work of [DC12] also leverages web-based nature of Helios' voting booth and the notion of divertible proofs (explained below) to provide a consensual vote copying mechanism in which Bob asks Alice for permission (and assistance) in copying her vote, so that the copied vote *cannot* be detected as such.

Blind signatures [Cha83, CP93], involve signing a document through an intermediary (in our case, the copier) without the original signer (the voter) being able to trace the end product. Blind signatures have been suggested by Chaum [Cha83] for use with anonymous electronic cash, where banks sign "coins" proving their authenticity but are unable to trace their use, and voting where authorities can supply signed blank ballots to authenticated voters but are then unable to track them once filled.

Divertible proofs [OO90, DGB88] are a similar notion to blind signatures, but in an online setting. An intermediate party is introduced between the prover and verifier, playing the role of the verifier against the prover and that of the prover against the verifier. The intermediate is called a *warden* in some cases (for example, if he is introduced to enforce to ensure honest behaviour) or a man in the middle in others.

In [SB13], Smyth and Bernhard correctly point out that the claim about Alice's privacy in [DC12] (i.e. *Bob will not learn anything about Alice's vote that is not also revealed by the tally, but will know that the ballot produced by the copying system contains the same vote as Alice's*) is too strong. Bob can learn more about Alice's vote than what is revealed by the tally, as he also knows if he cast his duplicate. Furthermore, they claim that vote copying is incompatible with ballot privacy. This is true if the definitions and security reductions do not account for ballot copying. We can however, adjust the definitions by adding a ballot copying oracle, and ensuring that it always copies votes from the board that will be tallied, even if the board that is visible to the adversary is the fake one.

2.3.3 Receipt Freeness and Coercion Resistance

The early definitions of receipt-freeness [BT94, Oka97] introduced the idea that an attacker should not be able to distinguish between a real transcript and a fake one, but their descriptions are rather informal. A weaker definition of receipt-freeness, proposed in [CG96, KZZ15], lets the attacker only interact with the voter *after* the election (in particular, the attacker cannot control the randomness of the voter's device). A simulation-based definition for receipt-freeness (UC-RF) was given in [MN06]. It assumes however that the voters adopt an "anti-coercion strategy" and is therefore closer to coercion-resistance as defined in [JCJ05], even if it does not cover, for instance, abstention attacks. The coercion-resistance and receipt-freeness definitions in [KT09] also assume a strategy from the voter.

Similarly, the symbolic definition of receipt-freeness in [DKR06] also requires the voter to adopt a strategy to fool the adversary. Other definitions in symbolic models aim at characterizing the notion of a receipt [JdV06, BG11, HS12] but (as usual in symbolic models) they are much more abstract than standard computational models.

In contrast, in chapter 9 we define *strong* receipt freeness which allows adversaries to have greater control over the voter. At the same time, we present BeleniosRF, which satisfies this definition without needing a specific counter-coercion strategy from the voter.

Previous receipt-free schemes. The scheme by Kiayias *et al.* [KZZ15] only achieves receipt-freeness for honest voters, as discussed above. Other well-known and deployed schemes include Prêt-à-voter [RBH$^+$09] and Scantegrity [CEC$^+$08]. These systems however are designed for elections with physical voting booths. The system used in Estonia [SFD$^+$14] and the one deployed in Norway [AC11, Gjø12] might possibly satisfy some level of receipt-freeness, as the corresponding ballot boxes are not publicly available. But because of this, they do not achieve universal verifiability.

Blazy et al. [BFPV11b] sketch a receipt-free e-voting scheme, using their new primitive of signatures on rerandomizable ciphertexts (SRC) which enable votes to be rerandomized without allowing them to be changed, although no definition nor security reduction for receipt-freeness is given. In their approach, the voting server, which is in charge of running the ballot box, can be allowed to re-randomise ballots if it is trusted not reveal the randomness used for that procedure.

In contrast, Helios [MBC01] and [KZZ15] achieve receipt-freeness under the assumption that the voting client is not going to reveal the randomness it used for sealing the vote. The latter seems difficult to ensure in practice, unless voters are provided with secure hardware tokens.

In chapter 9 we will present BeleniosRF which is receipt-free only relies on a single re-randomization server not leaking.

Chapter 3

Background & Definitions

3.1 Notation

We will use $y \leftarrow S$ to denote uniform sampling from set S. For an algorithm A, $y \leftarrow A(x;r)$ denotes the output of A on input x and randomness r. If the randomness is unimportant, we may use the notation $y \leftarrow A(x)$ which denotes that r is (implicitly) randomly sampled and passed to A. We write $y := f(x)$ when the value of y is derived as a deterministic function f of x.

We will assume that all algorithms get a security parameter λ, expressed in unary as an additional input. This is not only to make sure that the parameter is available, but to also enable us to make claims about the algorithms complexity with respect to the size of the security parameter. In general, we will consider algorithms to be efficient if their running time is polynomial in the security parameter.

We will call a function $f : \mathbb{N} \to \mathbb{R}$ *negligible*, if for any polynomial $p(t)$ there exists n_0 such that for $n > n_0$ we have $f(n) < p(n)$. We say that a probability a is overwhelming, if $1-a$ is negligible. Given two functions $g,h : \mathbb{N} \to [0,1]$ we write $g(n) \approx h(n)$ when $|g(n) - h(n)| = O(n^{-c})$ for every constant $c > 0$.

In many security definitions we ofter refer to probabilities of the form:

$$\Pr\Big[Exp(\lambda)\Big],$$

where $Exp(\lambda)$ describes an experiment involving the security parameter λ and a number of bound variables and algorithms. We will always consider the probability to be taken over the random sampling of such variables, and any implicit random sampling in the execution of algorithms during the experiment. As such, evaluating the probability produces a function in λ with range $[0,1]$. In the same context, we will sometimes consider constant values as (constant) functions of λ in order to facilitate comparison via the $\approx$ operator.

For algorithms A and B, $\langle A,B\rangle$ denotes algorithms A and B being run together, with messages passed between them during execution. We write $(x,y) \leftarrow \langle\!\langle A(a),B(b)\rangle\!\rangle$ where at the end A, with input a outputs x, whereas B, with input b outputs y. By $A^{B(b)}(a)$,we denote the invocation of B (with input b) by A (with input a). Note that A does not get the private output of B.

3.2 Setting

3.2.1 Cyclic Groups & The Discrete Logarithm Assumption

Let GGen be an algorithm that on input 1^λ returns $(\mathbb{G},p,g)$ such that $\mathbb{G}$ is the description of a finite cyclic group of prime order p, where $|p| = \lambda$, and g is a generator of $\mathbb{G}$.

Definition 1 (Discrete Logarithm Assumption)**.** *The discrete logarithm assumption holds relative to* GGen *if for all non-uniform polynomial time adversaries* $\mathcal{A}$

$$\Pr\Big[(\mathbb{G},p,g) \leftarrow \mathrm{GGen}(1^\lambda); h \leftarrow \mathbb{G}; a \leftarrow \mathcal{A}(\mathbb{G},p,g,h) : g^a = h\Big] \approx 0$$

In this definition, the value a is called the discrete logarithm of h in the basis g. Note that the discrete logarithm assumption is defined with respect to a particular group generator algorithm GGen. According to current state-of-the-art cryptanalytic techniques, to get a security level of $2^{-\lambda}$ the group generator may for example return well-chosen elliptic curve groups where group elements can be represented with $O(\lambda)$ bits or multiplicative subgroups of finite fields with a large characteristic where group elements can be represented with $O(\lambda^3)$ bits. It is well-known that the discrete logarithm assumption is equivalent to the following assumption.

Definition 2 (Discrete Logarithm Relation Assumption)**.** *For all* $n \geq 1$ *and all non-uniform polynomial time adversaries* $\mathcal{A}$

$$\Pr\left[\begin{array}{l}(\mathbb{G},p,g) \leftarrow \mathrm{GGen}(1^\lambda); g_1,\dots,g_n \leftarrow \mathbb{G}; \\ a_0,\dots,a_n \leftarrow \mathcal{A}(\mathbb{G},p,g,\{g_i\}_i)\end{array} : \exists a_i \neq 0 \text{ and } g^{a_0}\prod_{i=1}^{n} g_i^{a_i} = 1\right] \approx 0$$

We call such a product $g^{a_0}\prod_{i=1}^{n} g_i^{a_i} = 1$ a non-trivial discrete logarithm relation.

Definition 3 (Decisional Diffie-Hellman (DDH) Assumption)**.** *The DDH assumption holds relative to $\mathcal{G}$ if for all PPT adversaries $\mathcal{A}$*

$$\Pr\left[\begin{array}{c} gk = (\mathbb{G}, q, g) \leftarrow \mathrm{GGen}(1^\lambda); x, y, z \leftarrow \mathbb{Z}_p; b \leftarrow \{0,1\}; \\ h := g^x; u := g^y; v := g^{(1-b)xy+bz} : \mathcal{A}(gk, h, u, v) = b \end{array}\right] \approx \frac{1}{2}.$$

The DDH assumption relative to $\mathcal{G}$ implies the DL assumption relative to $\mathcal{G}$. The DDH assumption is believed to hold when $\mathbb{G}$ is an appropriately chosen subgroup of elliptic curve groups or multiplicative groups of large characteristic finite fields.

3.2.2 One-way functions

A one-way function f with setup $\mathcal{G}$ works as follows.

$\mathcal{G}(1^\lambda) \rightarrow gk$**:** The probabilistic polynomial time setup algorithm generates a setup gk. The setup gk defines a domain SK and a range VK, which have efficient algorithms for deciding membership and sampling elements.

$f(gk, sk) = vk$**:** A deterministic polynomial time function from the domain to the range, i.e., given $sk \in \mathrm{SK}$ we get $vk = f(gk, sk) \in \mathrm{VK}$. If $sk \notin \mathrm{SK}$, f returns an error symbol $\perp$.

Definition 4 (One-way)**.** *We require the one-way function to be hard to invert, i.e., for all probabilistic polynomial time adversaries $\mathcal{A}$*

$$\Pr\left[gk \leftarrow \mathcal{G}(1^\lambda); sk \leftarrow \mathrm{SK}; vk := f(gk, sk); sk' \leftarrow \mathcal{A}(gk, vk) : f(gk, sk') = vk\right] \approx 0.$$

We will instantiate f via group exponentiation, i.e. $x \mapsto g^x$ with domain $\mathbb{Z}_p$ and range $\mathbb{G}$. The one-wayness of f is then implied by the DL assumption.

3.3 Encrypting and Commiting

A non-interactive commitment scheme allows a sender to create a commitment to a secret value. She may later open the commitment and reveal the value in a verifiable manner. A commitment should be hiding, i.e., not reveal the secret value, and binding in the sense that a commitment cannot be opened to two different values.

Encryption can be thought of as a perfectly binding commitment scheme, with the additional property that knowledge of a secret key allows the receiver to obtain the original value without the need for the sender to reveal it.

3.3.1 Commitment Schemes

Formally, a non-interactive commitment scheme is a pair of probabilistic polynomial time algorithms $(\mathrm{CGen}, \mathrm{Com})$. The setup algorithm $ck \leftarrow \mathrm{CGen}(1^\lambda)$ generates a commitment key ck. The commitment key specifies a message space $\mathcal{M}_{ck}$, a randomness space $\mathcal{R}_{ck}$ and a commitment space $\mathcal{C}_{ck}$. The commitment algorithm combined with the commitment key specifies a function $\mathrm{Com}_{ck} : \mathcal{M}_{ck} \times \mathcal{R}_{ck} \to \mathcal{C}_{ck}$. Given a message $m \in \mathcal{M}_{ck}$ the sender picks uniformly at random $r \leftarrow \mathcal{R}_{ck}$ and computes the commitment $c = \mathrm{Com}_{ck}(m;r)$.

Definition 5 (Perfectly hiding)**.** *We say a non-interactive commitment scheme* $(\mathrm{CGen}, \mathrm{Com})$ *is perfectly hiding if a commitment does not reveal the committed value. For all non-uniform polynomial time stateful interactive adversaries* $\mathcal{A}$

$$\Pr\left[\begin{array}{l} ck \leftarrow \mathrm{CGen}(1^\lambda); (m_0, m_1) \leftarrow \mathcal{A}(ck); \\ b \leftarrow \{0,1\}; c \leftarrow \mathrm{Com}_{ck}(m_b) \end{array} : \mathcal{A}(c) = b \right] = \frac{1}{2}$$

where $\mathcal{A}$ *outputs* $m_0, m_1 \in \mathcal{M}_{ck}$.

Definition 6 (Binding)**.** *A non-interactive commitment scheme* $(\mathrm{CGen}, \mathrm{Com})$ *is computationally binding if a commitment can only be opened to one value. For all non-uniform polynomial time adversaries* $\mathcal{A}$

$$\Pr\left[\begin{array}{l} ck \leftarrow \mathrm{CGen}(1^\lambda); \\ (m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}(ck) \end{array} : \begin{array}{c} \mathrm{Com}_{ck}(m_0; r_0) = \mathrm{Com}_{ck}(m_1; r_1) \\ \text{and } m_0 \neq m_1 \end{array} \right] \approx 0$$

where $\mathcal{A}$ *outputs* $m_0, m_1 \in \mathcal{M}_{ck}$ *and* $r_0, r_1 \in \mathcal{R}_{ck}$.

We say a commitment scheme is homomorphic if for all valid keys ck the message, randomness and commitment spaces are abelian groups and for all messages $m_0, m_1 \in \mathcal{M}_{ck}$ and randomness $r_0, r_1 \in \mathcal{R}_{ck}$ we have

$$\mathrm{Com}_{ck}(m_0; r_0) \cdot \mathrm{Com}_{ck}(m_1; r_1) = \mathrm{Com}_{ck}(m_0 + m_1; r_0 + r_1).$$

3.3.2 Pedersen Commitments

The most prominent example of a homomorphic perfectly hiding commitment scheme is the Pedersen commitment scheme [Ped91]. Pedersen commitments have the form $c = g^r h^m$ where g, h are group elements specified in the commitment key. The opening of a Pedersen commitment is $(m,r) \in \mathbb{Z}_p^2$, from which anybody can recompute the commitment c and verify it was a valid commitment. Since Pedersen commitments are random group elements, they are perfectly hiding. On the other hand, breaking the binding property of Pedersen commitments corresponds to breaking the discrete logarithm assumption.

$\mathrm{CGen}(gk)$	$\mathrm{Com}_{ck}(m_1,\ldots,m_n)$
$h_1,\ldots,h_n \leftarrow \mathbb{G}$.	If $(m_1,\ldots,m_n) \notin \mathbb{Z}_p^n$ return $\perp$.
Return $ck := (h_1,\ldots,h_n)$.	$r \leftarrow \mathbb{Z}_p$; Return $c := g^r \prod_{i=1}^{n} h_i^{m_i}$.

Figure 3.1: Pedersen commitment to multiple elements

We exploit the fact that the Pedersen commitment scheme is homomorphic, i.e., for all correctly generated gk, ck and all $m, m', r, r' \in \mathbb{Z}_p$

$$\mathrm{Com}_{ck}(m;r) \cdot \mathrm{Com}_{ck}(m';r') = \mathrm{Com}_{ck}(m+m';r+r').$$

We will use a variant of the Pedersen commitment scheme to commit to multiple messages at once as shown in Fig. 3.1.

With the Pedersen commitment scheme in mind, we will assume throughout the text that the message space is $\mathbb{Z}_p^n$ and the randomness space is $\mathbb{Z}_p$. The constructions we have in Sections 5.4 and 5.6.3 require a perfectly hiding, homomorphic commitment scheme so we are not limited to using the Pedersen commitment scheme. However, in Sections 5.5 and 5.6.4, we will rely on specific properties of the Pedersen scheme and work directly on the group elements in the key.

3.3.3 Public-Key Encryption

A public-key encryption scheme (over setup gk) consists of three algorithms $(\mathrm{PKEGen}, \mathcal{E}, \mathcal{D})$. $\mathrm{PKEGen}(gk)$ is a probabilistic algorithm that generates a public key and decryption key pair (pk, dk). Without loss of generality, we assume pk can be efficiently computed given dk and write $pk = \mathrm{PKEGen}(gk, dk)$ for this computation which returns $\perp$ if dk is not valid. $\mathcal{E}(pk, m)$ is a probabilistic algorithm which returns a ciphertext c if all its inputs are valid and $\perp$ otherwise. $\mathcal{D}(dk, c)$ is a deterministic algorithm that decrypts the ciphertext and returns either the message m or the failure symbol $\perp$.

We assume that gk, which is an implicit input to $\mathcal{E}$ and $\mathcal{D}$, defines the public key, decryption key, message, randomness and ciphertext spaces PK_{gk}, DK_{gk}, $\mathcal{M}_{gk}$, $\mathcal{R}_{gk}$ and $\mathcal{C}_{gk}$.

In most chapters of this work, we will make use of an encryption scheme where the message space is $\mathbb{Z}_p$ for some large integer p, which is explicitly or implicitly defined by the public key, and with size $|p| = \ell_p(\lambda)$ for a publicly known polynomial ℓ_p.

For notational convenience, we let $c^{\vec{z}}$ be the vector $(c^{z_1}, \dots, c^{z_n})$ given a ciphertext c and a vector of integers $\vec{z} = (z_1, \dots, z_n)$. Given a vector $\vec{w}$ we also define $\vec{c} \leftarrow \mathcal{E}_{ek}(\vec{w})$ as the vector of ciphertexts given by $(\mathcal{E}_{ek}(w_1), \dots, \mathcal{E}_{ek}(w_n))$.

3.3.4 IND-CPA Security

A common security baseline for encryption is *indistinguishability under chosen plaintext attacks (IND-CPA).*

Definition 7 (IND-CPA security)**.** *We say that* (PKEGen, $\mathcal{E}, \mathcal{D}$) *is indistinguishable under chosen plaintext attack (IND-CPA secure) if for all probabilistic polynomial time stateful adversaries $\mathcal{A}$*

$$\Pr\left[\begin{array}{l} gk \leftarrow \mathcal{G}(1^\lambda); (pk, dk) \leftarrow \text{PKEGen}(gk) \\ (m_0, m_1) \leftarrow \mathcal{A}(gk, pk); b \leftarrow \{0,1\}; c \leftarrow \mathcal{E}(pk, m_b) \end{array} : \mathcal{A}(c) = b\right] \approx \frac{1}{2},$$

where $\mathcal{A}$ outputs $m_0, m_1 \in \mathcal{M}_{gk}$.

3.3.5 ElGamal Encryption

In many of the following chapters we will use ElGamal encryption, as described in Fig. 3.2, which is IND-CPA secure if the DDH assumption holds relative to GGen where $gk = (\mathbb{G}, q, g) \leftarrow \text{GGen}(1^\lambda)$.

We have $\text{PK}_{gk} := \mathbb{G}^*$, $\text{DK}_{gk} := \mathbb{Z}_q^*$, $\mathcal{M}_{gk} := \mathbb{G}$, $\mathcal{R}_{gk} := \mathbb{Z}_p$, and $\mathcal{C}_{gk} := \mathbb{G}^2$. We also note that ElGamal ciphertexts are homomorphic, similarly to Pedersen commitments.

PKEGen(gk)	$\mathcal{E}(pk, m)$	$\mathcal{D}(dk, c = (u, v))$
$dk \leftarrow \mathbb{Z}_q^*; pk := g^{dk}$.	If $pk \notin \mathbb{G}^*$ or $m \notin \mathbb{G}$ return $\perp$.	If $dk \notin \mathbb{Z}_q^*$ or $c \notin \mathbb{G}^2$ return $\perp$.
Return (pk, dk).	$r \leftarrow \mathbb{Z}_p$; Return $c := (pk^r, g^r m)$.	Return $m := vu^{-\frac{1}{dk}}$.

Figure 3.2: ElGamal encryption

3.3.6 IND-CCA Security

A common security baseline for encryption is *indistinguishability under chosen ciphertext attacks (IND-CCA).*

Definition 8 (IND-CCA security)**.** *We say that* (PKEGen, $\mathcal{E}, \mathcal{D}$) *is indistinguishable under chosen ciphertext attack (IND-CCA secure) if for all probabilistic polynomial time stateful* *adversaries $\mathcal{A}$*

$$\Pr\left[\begin{array}{l} gk \leftarrow \mathcal{G}(1^\lambda); (pk,dk) \leftarrow \text{PKEGen}(gk) \\ (m_0,m_1) \leftarrow \mathcal{A}^{\mathcal{D}}(gk,pk); b \leftarrow \{0,1\}; c \leftarrow \mathcal{E}(pk,m_b) \end{array} : \mathcal{A}^{\mathcal{D}^*}(c) = b\right] \approx \frac{1}{2},$$

where $\mathcal{A}$ outputs $m_0, m_1 \in \mathcal{M}_{gk}$, and $\mathcal{D}^*$ is defined as $\mathcal{D}$ except it answers $\perp$ when queried on c.

The above definition follows the more recent (adaptive) CCA2 [RS91] variant of chosen ciphertext attacks. We can recover the older CCA1 definition by omitting the $\mathcal{D}*$ oracle, but will not be using it in this text.

3.4 Zero Knowledge

3.4.1 Relations and NP-Languages

Let R be a polynomial time decidable binary relation, i.e., a relation that defines a language in NP. We call w a witness for a statement u if $(u,w) \in R$. We define the language

$$L_R = \{x \mid \exists w : (x,w) \in R\}$$

as the set of statements x that have a witness w in the relation R.

To incorporate the security parameter into relations, we will without loss of generality assume all statements are of a form such that λ can be easily derived (all statements in this text could be reformulated to be of the form $u = (1^\lambda, u')$ although for notational convenience we will not do this) and that all statements and witnesses are of size polynomial in λ.

3.4.2 Interactive Protocols

A zero knowledge protocol for an NP-relation R enables a prover to demonstrate to a verifier that a statement u satisfies $u \in L_R$, i.e. that there exists a witness w such that $(u,w) \in R$ without disclosing anything else, in particular not disclosing the value of w that the prover has in mind.

More formally, a protocol for a relation R w.r.t. a setup $gk \leftarrow \text{GGen}(1^\lambda)$ is a tuple $(\text{CRSGen}, \mathcal{P}, \mathcal{V})$. If a common reference string, crs is used, we assume that the generator, $\text{CRSGen}(gk)$ generates it before the execution of the protocol. Both gk and crs are provided as implicit inputs to all algorithms. We consider that the prover $\mathcal{P}$ and the verifier $\mathcal{V}$, are both probabilistic polynomial time interactive algorithms. The transcript produced by $\mathcal{P}$ and

$\mathcal{V}$ when interacting on inputs s and t is denoted by $tr \leftarrow \langle \mathcal{P}(s), \mathcal{V}(t) \rangle$. In this text, the prover will not produce an output (apart from messages to the verifier), so, for convenience we only consider the output of the verifier and write $b \leftarrow \langle\!\langle \mathcal{P}(s), \mathcal{V}(t) \rangle\!\rangle$ depending on whether the verifier rejects, $b = 0$, or accepts, $b = 1$. Without loss, we assume that transcripts include the outputs and call them *accepting* or *rejecting* depending on whether b is 1 or not.

Σ-protocols are a specific kind of 3-move interactive protocols which are *public coin*, *complete*, *2-special-sound* and have special honest verifier zero knowledge. A typical run of a Σ-proocol is illustrated in Fig. 3.3. $\mathcal{P}$ generates an initial message a; $\mathcal{V}(\mathsf{crs}, u, a)$, produces a random challenge x; $\mathcal{P}(x)$ computes a response z to the verifier‘s random challenge x. Finally, $\mathcal{V}(\mathsf{crs}, u, a, x, z)$ verifies the proof and outputs 1 for acceptance or 0 for rejection.

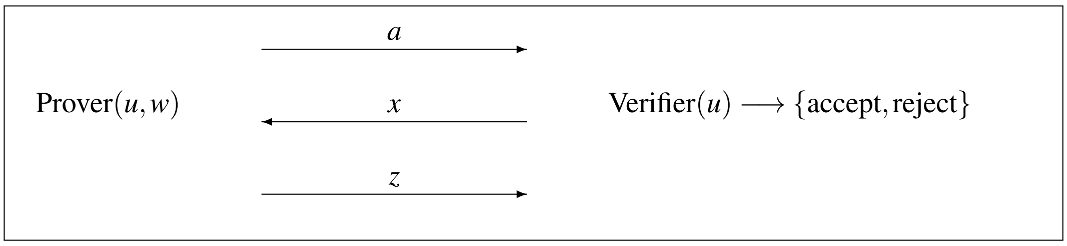

a
Prover(u, w)
x
Verifier$(u) \longrightarrow \{\text{accept}, \text{reject}\}$
z

Figure 3.3: Σ-protocol with statement u and witness w

A Σ-protocol is required to operate correctly when used by honest participants (*completeness*), to prevent dishonest provers from convincing verifiers that false statements hold (*soundness*), and not to leak information about w (*zero-knowledge*).

Definition 9 (Public coin). *A protocol $(\mathcal{P}, \mathcal{V})$ is called* public coin *if the verifier chooses his messages uniformly at random and independently of the messages sent by the prover, i.e., the challenges correspond to the verifier's randomness ρ.*

Definition 10 (Perfect completeness). *$(\mathcal{P}, \mathcal{V})$ has perfect completeness if for all non-uniform polynomial time adversaries $\mathcal{A}$*

$$\Pr\left[(u, w) \leftarrow \mathcal{A}(1^\lambda) : (u, w) \notin R \text{ or } \langle\!\langle \mathcal{P}(x, w), \mathcal{V}(x) \rangle\!\rangle = 1\right] = 1$$

Definition 11 (n-Special Soundness). *We say that $(\mathcal{P}, \mathcal{V})$ is n-special sound [GK14] if there exists an efficient extractor χ such that for any statement u, given n accepting transcripts $\{(a, x_i, z_i)\}_{i=1}^{n}$ where the challenges x_i are distinct, $\chi(u, a, x_i, z_i)$ outputs w s.t. $(u, w) \in \mathcal{R}$.*

A protocol is zero-knowledge if it does not leak information about the witness beyond what can be inferred from the truth of the statement. We will present protocols that have special honest verifier zero-knowledge in the sense that if the verifier's challenges are known in advance, then it is possible to simulate the entire argument without knowing the witness.

Definition 12 (Special Honest Verifier Zero-Knowledge (SHVZK))**.** *A public coin argument $(\mathcal{P},\mathcal{V})$ is called a* special honest verifier zero knowledge (SHVZK) *argument for R if there exists a probabilistic polynomial time simulator $\mathcal{S}$ such that for all interactive non-uniform polynomial time adversaries $\mathcal{A}$*

$$\Pr\Big[(u,w,\rho)\leftarrow\mathcal{A}(1^\lambda); tr\leftarrow\langle\mathcal{P}(u,w),\mathcal{V}(u;\rho)\rangle : (u,w)\in R \text{ and } \mathcal{A}(tr)=1\Big]$$

$$\approx \Pr\Big[(u,w,\rho)\leftarrow\mathcal{A}(1^\lambda); tr\leftarrow\mathcal{S}(u,\rho) : (u,w)\in R \text{ and } \mathcal{A}(tr)=1\Big]$$

where ρ is the public coin randomness used by the verifier.

If this holds also for unbounded adversaries $\mathcal{A}$, we say $(\mathcal{P},\mathcal{V})$ is statistically special honest verifier zero-knowledge. If the two distributions are equal, then we say $(\mathcal{P},\mathcal{V})$ is perfect special honest verifier zero-knowledge.

3.4.3 Zero Knowledge Arguments of Knowledge

In some applications we may need to tackle more complex protocols, where n-special soundness may not be simple to obtain. Arguments of knowledge give us an appropriate framework to analyse the security of such protocols.

To define an argument of knowledge we follow Groth and Ishai [GI08] that borrowed the term witness-extended emulation from Lindell [Lin03]. Informally, their definition says that given an adversary that produces an acceptable argument with some probability, there exists an emulator that produces a similar argument with the same probability together with a witness w. Note that the emulator is allowed to rewind the prover and verifier's interaction to any previous move.

Definition 13 (Argument of knowledge)**.** *The pair $(\mathcal{P},\mathcal{V})$ is called an* argument of knowledge *for the relation R if we have perfect completeness and statistical witness-extended emulation as defined below.*

Definition 14 (Statistical witness-extended emulation)**.** *$(\mathcal{P},\mathcal{V})$ has* statistical witness-extended emulation *if for all deterministic polynomial time $\mathcal{P}^*$ there exists an expected* *polynomial time emulator $\mathcal{E}$ such that for all interactive adversaries $\mathcal{A}$*

$$\Pr\left[(u,s) \leftarrow \mathcal{A}(1^\lambda); tr \leftarrow \langle \mathcal{P}^*(u,s), \mathcal{V}(u)\rangle : \mathcal{A}(tr) = 1\right]$$
$$\approx \Pr\left[\begin{array}{l} (u,s) \leftarrow \mathcal{A}(1^\lambda); (tr,w) \leftarrow \mathcal{E}^{\langle \mathcal{P}^*(u,s), \mathcal{V}(u)\rangle}(u) : \\ \mathcal{A}(tr) = 1 \text{ and if } tr \text{ is accepting then } (u,w) \in R \end{array}\right]$$

where the oracle called by $\mathcal{E}^{\langle \mathcal{P}^*(u,s), \mathcal{V}(u)\rangle}$ allows rewinding i.e. the oracle is aware of the operation of both algorithms and thus the execution of $\langle \mathcal{P}^*(u,s), \mathcal{V}(u)\rangle$ may be paused at any intermediate state, then resumed from such state as requested. Additionally, when resuming, any part of the verifier randomness contained in s that has not yet been read, can be replaced with fresh randomness.

In the definition, s can be interpreted as the state of $\mathcal{P}^*$, including the randomness. So, whenever $\mathcal{P}^*$ is able to make a convincing argument when in state s, $\mathcal{E}$ can extract a witness. This is why we call it an argument of knowledge.

The following lemma links witness-extended emulation with n-special soundness and its generalizations.

3.4.4 A general forking lemma

Suppose that we have a $(2\mu+1)$-move public-coin argument with μ challenges, $x_1, \ldots, x_\mu$ in sequence. Let $n_i \geq 1$ for $1 \leq i \leq \mu$. Consider $\prod_{i=1}^{\mu} n_i$ accepting transcripts with challenges in the following tree format. The tree has depth μ and $\prod_{i=1}^{\mu} n_i$ leaves. The root of the tree is labelled with the statement. Each node of depth $i < \mu$ has exactly n_i children, each labelled with a distinct value for the ith challenge x_i.

This can be referred to as an $(n_1, \ldots, n_\mu)$-tree of accepting transcripts. All of our arguments allow a witness to be extracted efficiently from an appropriate tree of accepting transcripts. This is a natural generalisation of special-soundness for Sigma-protocols, where $\mu = 1$ and $n = 2$. For simplicity in the following lemma, we assume that the challenges are chosen uniformly from $\mathbb{Z}_p$ where $|p| = \lambda$, but any sufficiently large challenge space would suffice.

Lemma 1 (Forking Lemma). *Let $(\mathcal{P}, \mathcal{V})$ be a $(2\mu+1)$-move, public coin interactive protocol. Let χ be a witness extraction algorithm that always succeeds in extracting a witness from an $(n_1, \ldots, n_\mu)$-tree of accepting transcripts in probabilistic polynomial time. Assume that $\prod_{i=1}^{\mu} n_i$ is bounded above by a polynomial in the security parameter λ. Then $(\mathcal{P}, \mathcal{V})$*

has witness-extended emulation.

For simplicity in the following proof, we assume that the challenges are chosen uniformly from $\mathbb{Z}_p$ where $|p| = \lambda$, but any sufficiently large challenge space would suffice.

Proof. Suppose that for deterministic polynomial time $\mathcal{P}^*$ there is a non-uniform polynomial time interactive adversary $\mathcal{A}$ in the sense of witness-extended emulation, such that

$$\Pr\left[(u,s) \leftarrow \mathcal{A}(1^\lambda); tr \leftarrow \langle \mathcal{P}^*(u,s), \mathcal{V}(u)\rangle : \mathcal{A}(tr) = 1\right] = \varepsilon.$$

Note that if ε is negligible, then we do not need to extract a witness, since the emulator can simply fail every time and trivially achieve witness-extended emulation. Therefore, from now on, we assume that ε is not negligible.

We construct an expected polynomial time emulator $\mathcal{U}$, which has access to a rewindable transcript oracle $\langle \mathcal{P}^*, \mathcal{V}\rangle$ and produces a witness. This is done via recursive calls to tree-finders $\mathcal{T}$ that deal with the protocol after the first few challenges are already fixed. The ith tree-finder takes the previous challenges and partial transcript given to it as input, picks random values for x_{i+1}, runs the prover on these values and hands the result to the next tree-finder. Each tree-finder may fail on the first value of x_{i+1}, ensuring that the whole process runs in expected polynomial time. With overwhelming probability, the emulator obtains an $(n_1, \ldots, n_\mu)$-tree of transcripts and is then able to extract a witness, using the efficient algorithm χ that exists by assumption.

```
$\mathcal{U}^{\langle \mathcal{P}^*, \mathcal{V}\rangle}(u) \rightarrow (tr, w)$:
    -Run $\mathcal{T}^{\langle \mathcal{P}^*, \mathcal{V}\rangle}(1) \rightarrow (tr, \mathsf{tree})$
    -If $\mathsf{tree} = \perp$ then return $(tr, \perp)$.
    -If $\mathsf{tree}$ is not a valid $(n_1, \ldots, n_\mu)$-tree of transcripts (i.e. there are collisions
    in certain challenges) then return $(tr, \perp)$.
    -Else run $w \leftarrow \chi(u, \mathsf{tree})$.
    -Return $(tr, w)$
```

Figure 3.4: Emulator $\mathcal{U}$ for Lemma 1

The emulator $\mathcal{U}$ calls $\mathcal{T}^{\langle \mathcal{P}^*, \mathcal{V}\rangle}(i)$ for $1 \le i \le \mu + 1$:

Fix $1 \le i \le \mu$, and fix $x_1, \ldots, x_{i-1}$. We say that $\mathcal{T}(i)$ has failed if it returns $(tr, \perp)$.

Let ε' be the probability that the $\mathcal{T}(i)$ fails for this choice of challenges, and let $\varepsilon'(x_i)$ be the probability that $\mathcal{T}(i+1)$ fails for this choice of challenges continued with x_i. The

```
$\mathcal{T}^{\langle \mathcal{P}^*,\mathcal{V}\rangle}(i) \rightarrow (tr, \mathsf{tree})$:
    -If $i = \mu + 1$
        -Obtain a complete protocol transcript from $tr \leftarrow \langle \mathcal{P}^*,\mathcal{V}\rangle$
        -Run $\mathcal{V}(tr) \rightarrow b$
        -If $b = 0$ then return $(tr, \perp)$.
        -If $b = 1$ then set $\mathsf{tree} = \{tr\}$ and return $(tr, \mathsf{tree})$.
    -Run $\langle \mathcal{P}^*,\mathcal{V}\rangle$ up to and including move $2i+1$.
    -Run $\mathcal{T}^{\langle \mathcal{P}^*,\mathcal{V}\rangle}(i+1) \rightarrow (tr, \mathsf{tree})$
    -If $\mathsf{tree} = \perp$ then return $(tr, \perp)$.
    -Set $\mathsf{counter} = 1$
    -While $\mathsf{counter} < n_i$:
        -Rewind $\langle \mathcal{P}^*,\mathcal{V}\rangle$ back until just before move $2i$.
        -Run $\mathcal{T}^{\langle \mathcal{P}^*,\mathcal{V}\rangle}(i+1) \rightarrow (tr', \mathsf{tree}')$
        -If $\mathsf{tree} \neq \perp$, then append the transcripts in $\mathsf{tree}'$ to $\mathsf{tree}$, and increment
    $\mathsf{counter}$.
    -Return $(tr, \mathsf{tree})$
```

Figure 3.5: Tree finders $\mathcal{T}$ for Lemma 1

ith tree-finder can fail only if the $(i+1)$th tree-finder fails the first time it is called. This implies that for uniformly random x_i, the probability that $\mathcal{T}(i+1)$ fails is $\varepsilon' = \sum_{x_i \in \mathbb{Z}_p} \Pr[X = x_i]\varepsilon'(x_i)$.

Therefore, the expected number of times that $\mathcal{T}(i)$ runs $\mathcal{T}(i+1)$ is $1 + \varepsilon' \frac{(n_i - 1)}{\varepsilon'} = n_i$. The final tree-finder $\mathcal{T}(k+1)$ merely checks whether the transcript is accepting or not. Hence, the total expected running time for $\mathcal{T}(1)$ to be $\prod_{i=1}^{\mu} n_i$ multiplied by the time taken to check whether a transcript is accepting. We conclude that the emulator $\mathcal{U}$ runs in expected polynomial time.

The first tree-finder $\mathcal{T}(1)$ only outputs $(tr, \perp)$ if the very first set of challenges generated by all of the emulators fails to produce an accepting transcript. This is exactly the probability that $\mathcal{P}^*$ successfully produces an accepting transcript in one run.

Given that we receive $\prod_{i=1}^{\mu} n_i$ accepting transcripts in tree, we now consider the probability that they do not form an $(n_1, \ldots, n_\mu)$-tree. This occurs only when the n_i values of challenge x_i used by $\langle \mathcal{P}^*,\mathcal{V}\rangle$ while in the loop controlled by $\mathsf{counter}$ are not distinct, or in other words, there is a collision between these values, for some i.

By Markov's inequality, an algorithm whose expected running time is t will only run for longer than time $T > t$ with probability $\frac{t}{T}$. Let t be the expected running time of $\mathcal{U}$, which is bounded above by a polynomial in the security parameter. For easier analysis, we limit the actual running time of $\mathcal{U}$ to T, whose value will be chosen later.

When $\mathcal{U}$ runs in time at most T, then at most T uniformly random public coin challenges were selected by $\mathcal{V}$ in $\langle \mathcal{P}^*, \mathcal{V} \rangle$. If there are no collisions between *any* of the public coins chosen, then there are certainly no collisions of the type which would prevent tree from being a $(n_1, \ldots, n_\mu)$-tree of transcripts. The probability that there is a collision between T values sampled uniformly from $\mathbb{Z}_p$ is at most $\frac{T^2}{p}$.

Now, we choose $T = \sqrt[3]{p}$. The probability that tree fails to be an $(n_1, \ldots, n_\mu)$-tree is at most $\frac{t}{T} + \frac{T^2}{p}$ which is now equal to $\frac{t}{\sqrt[3]{p}} + \frac{1}{\sqrt[3]{p}}$. This is negligible. Therefore, there is negligible probability of the tree-finding algorithms succeeding, yet $\mathcal{U}$ failing to extract a witness. This proves the argument has statistical witness-extended emulation. $\square$

3.4.5 Non-Interactive Zero-Knowledge (NIZK) Proofs

It is often desirable to operate in a single step, avoiding the interaction needed to execute a Σ-protocol. The prover still wishes to demonstrate to the verifier the truth of a statement $u \in L_R$ for an NP-relation R without disclosing any other information about her witness w.

A NIZK proof system (over a setup gk) for an NP-relation $\mathcal{R}$ defining the language $\mathcal{L}_\mathcal{R} := \{u \mid \exists w : (u, w) \in \mathcal{R}\}$, where u is a statement and w is a witness, is a tuple of polynomial-time algorithms $(\text{CRSGen}, \mathsf{Prove}, \text{PVfy})$. $\text{CRSGen}(gk)$ generates a common reference string crs; $\mathsf{Prove}(\mathsf{crs}, u, w)$ returns a proof π that $(u, w) \in \mathcal{R}$; $\text{PVfy}(\mathsf{crs}, u, \pi)$ verifies that π is a valid proof for $u \in \mathcal{L}_\mathcal{R}$, outputting a bit accordingly.

Perfect completeness of the proof system requires that for any *crs* generated by CRSGen and any pair $(u, w) \in \mathcal{R}$ we have $\Pr[\text{PVfy}(\mathsf{crs}, u, \mathsf{Prove}(\mathsf{crs}, u, w))] = 1$. Additionally, we require *soundness* and *zero-knowledge*, which are as follows:

- Soundness: For all PPT adversaries $\mathcal{A}$, we have

$$\Pr\left[\begin{array}{c} gk \leftarrow \text{GGen}(1^\lambda); \mathsf{crs} \leftarrow \text{CRSGen}(gk); (u, \pi) \leftarrow \mathcal{A}(gk, \mathsf{crs}) : \\ \text{PVfy}(\mathsf{crs}, u, \pi) = 1 \ \wedge \ u \notin \mathcal{L}_\mathcal{R} \end{array}\right] \approx 0.$$

- Zero-Knowledge: There exist PPT algorithms $(\text{SimCRSGen}, \text{SimProve})$, where $\text{SimCRSGen}(gk)$ outputs a simulated reference string crs and possibly a simulation trapdoor τ, and $\text{SimProve}(\mathsf{crs}, s, \tau)$ produces a simulated proof (without knowing a witness). We require that

$$\begin{aligned}&\Pr\left[gk \leftarrow \mathrm{GGen}(1^\lambda); \mathsf{crs} \leftarrow \mathrm{CRSGen}(gk) : \mathcal{A}^{\mathsf{Prove}}(gk, \mathsf{crs}) = 1\right]\\ \approx &\Pr\left[gk \leftarrow \mathrm{GGen}(1^\lambda); (\mathsf{crs}, \tau) \leftarrow \mathrm{SimCRSGen}(gk) : \mathcal{A}^{\mathsf{Sim}}(gk, \mathsf{crs}) = 1\right],\end{aligned}$$

where on query $(u, w) \in \mathcal{R}$, Sim returns $\pi \leftarrow \mathrm{SimProve}(\mathsf{crs}, u, \tau)$.

3.4.6 The Fiat-Shamir heuristic.

The Fiat-Shamir transformation takes an interactive public coin argument and replaces the challenges with the output of a cryptographic hash function H. The idea is that the hash function will produce random looking output and therefore be a suitable replacement for the verifier. The Fiat-Shamir heuristic yields a non-interactive zero-knowledge argument in the random oracle model [BR93].

Given a Σ-protocol $\mathcal{P}, \mathcal{V}$ where the verifier's challenge is sampled from X, and a hash function $h : \{0,1\}^* \to X$. the Fiat-Shamir transformation replaces the second step, $x \leftarrow V(\mathsf{crs}, u, a)$ with $x := h(\mathsf{crs}, u, a)$. As h is deterministic, $\pi \leftarrow \langle \mathcal{P}, h \rangle$ can be produced by $\mathcal{P}$ alone. At the same time, the original verifier $\mathcal{V}$ can be used unmodified to check π.

In general, the transformation allows us to transform any complete, public coin, special sound, SHVZK interactive protocol $(\mathrm{CRSGen}, \mathcal{P}, \mathcal{V})$ into a NIZK proof system $(\mathrm{CRSGen}, \langle \mathcal{P}, h \rangle, \mathcal{V})$ given an appropriate[1] hash function h.

In security proofs under the random oracle model, the hash function used in the protocol is assumed to be an oracle under the control of the simulator. As such, the simulator can choose the value returned by the hash function on any input with the only limitation being consistency (i.e. after setting $H(x) = y$, the simulator is not allowed to set $H(x) = y' \neq y$. The random oracle model has been criticised as [GK03, CGH04] have shown that it is possible to construct protocols that are secure under the random oracle model but provably insecure in general. At the same time, these results have not led to a vulnerability being found in a currently used protocol. In Chapter 4 we examine the DFN transformation that can be used as an alternative to Fiat-Shamir for a restricted class of protocols.

3.4.7 Signature of Knowledge

A *Signature of Knowledge* (SoK) for an NP-relation $\mathcal{R}$ w.r.t. a setup gk is a tuple $(\mathsf{SoKSetup}, \mathsf{SoKSign}, \mathsf{SoKVerify})$. $\mathrm{SoKSetup}(gk)$ outputs public parameters crs;

[1] For notational convenience we assume that the challenge space is the same for all rounds.

$\mathrm{SoKSign}(\mathsf{crs}, u, w, m)$ outputs a signature σ_{SoK} on m if $(u, w) \in \mathcal{R}$; $\mathrm{SoKVerify}(\mathsf{crs}, u, m, \sigma_{\mathrm{SoK}})$ outputs 1 if σ_{SoK} is a valid signature on m or 0 otherwise. The (game-based) security definition for signatures of knowledge (*SimExt*) [CL06], besides correctness, requires *Simulatability* and *Extractability*. We consider a *stronger* generalisation of the latter called f-extractability [BCKL08]:

- Simulatability: There are PPT algorithms $(\mathrm{SoKSimSetup}, \mathrm{SoKSimSign})$, where $\mathsf{SoKSimSetup}(gk)$ outputs public parameters crs and some trapdoor τ, whereas $\mathrm{SoKSimSign}(\mathsf{crs}, \tau, u, m)$ outputs a signature σ_{SoK}, such that

$$\begin{aligned}&\Pr\left[gk \leftarrow \mathrm{GGen}(1^\lambda); (\mathsf{crs}, \tau) \leftarrow \mathsf{SoKSimSetup}(gk) : \mathcal{A}^{\mathrm{SoKSim}}(gk, \mathsf{crs}) = 1\right]\\ \approx &\Pr\left[gk \leftarrow \mathrm{GGen}(1^\lambda); \mathsf{crs} \leftarrow \mathrm{SoKSetup}(gk) : \mathcal{A}^{\mathrm{SoKSign}}(gk, \mathsf{crs}) = 1\right],\end{aligned}$$

  for all PPT adversaries $\mathcal{A}$, where $\mathrm{SoKSim}(u, w, m)$ returns $\mathrm{SoKSimSign}(\mathsf{crs}, \tau, u, m)$ if $(u, w) \in \mathcal{R}$ and $\perp$ otherwise.

- f-Extractability: For all PPT adversaries $\mathcal{A}$, there exists a polynomial time algorithm $\mathsf{SoKExtract}$ such that:

$$\Pr\left[\begin{array}{l} gk \leftarrow \mathrm{GGen}(1^\lambda); (\mathsf{crs}, \tau) \leftarrow \mathsf{SoKSimSetup}(gk);\\ (u, m, \sigma_{\mathrm{SoK}}) \leftarrow \mathcal{A}^{\mathrm{SoKSim}}(gk, \mathsf{crs});\\ y \leftarrow \mathsf{SoKExtract}(\mathsf{crs}, \tau, u, m, \sigma_{\mathrm{SoK}}) :\\ \quad (u, m, \sigma_{\mathrm{SoK}}) \in Q_{\mathrm{SoKSim}} \vee \ \mathsf{SoKVerify}(\mathsf{crs}, u, m, \sigma_{\mathrm{SoK}}) = 0\\ \quad \vee \left(\exists w \text{ s.t. } (u, w) \in \mathcal{R} \wedge y = f(w)\right) \end{array}\right] \approx 1\cdot$$

  In the above, Q_{SoKSim} is a list of queries to the SoKSimSign oracle. Note that our extractability definition is stronger than that of [CL06], as we allow the adversary to ask for signatures w.r.t. statements for which it does know the witness. In the definition, if f is the identity function, we get the standard notion of extractability.

Signatures of knowledge in the random oracle model can be efficiently realised by applying the Fiat-Shamir transformation to Σ-protocols. Applying the transformation to Σ-protocols having quasi-unique responses (i.e. given an accepting transcript, it is infeasible to find a different accepting response w.r.t. the same initial message and challenge) provides weak simulation-extractability [FKMV12], where the extractor needs to rewind the prover. To get straightline f-extractability, i.e. without rewinding [Fis05], we additionally encrypt a function f of the witness with a public key in the reference string and prove that the encrypted value is consistent with the witness. This way we get both full weak extractability and straightline f-extractability simultaneously.

3.4.8 Non-Interactive Designated Verifier Zero Knowledge Arguments

Designated Verifier protocols have an additional property: they are tailored to a specific verifier as opposed to the general public. Conceptually, it is akin to each verifier having a personal CRS (along with additional related information that enables verification).

In a non-interactive designated verifier zero knowledge argument system, we imagine the verifier sets up a public key pk for the proof together with a secret verification key vk that can be used to verify the arguments. The system therefore consists of three probabilistic polynomial time algorithms $(\mathsf{RKGen}, \mathcal{P}, \mathcal{V})$.

$(pk, vk) \leftarrow \mathcal{G}(1^\lambda)$**:** The key generation algorithm, given the security parameter as input, generates a public key pk and a secret verification key vk.

$\pi \leftarrow \mathcal{P}(pk, u, w)$**:** Given a public key pk and $(u, w) \in R_\lambda$, the prover algorithm generates an argument π.

$\{0,1\} \leftarrow \mathcal{V}(vk, u, \pi)$**:** Given a secret verification key vk, a statement u and an argument π, the verification algorithm returns 1 if accepting the argument and 0 for rejection of the argument.

$(\mathsf{RKGen}, \mathcal{P}, \mathcal{V})$ is said to be a non-interactive designated verifier zero-knowledge argument system for R with culpable soundness with respect to R_{guilt} if it is complete, culpably sound and zero-knowledge as defined below.

Definition 15 (Completeness)**.** $(\mathsf{RKGen}, \mathcal{P}, \mathcal{V})$ *is perfectly complete if for all* $\lambda \in \mathbb{N}$ *and all* $(u, w) \in R_\lambda$

$$\Pr\left[(pk, vk) \leftarrow \mathcal{G}(1^\lambda); \pi \leftarrow \mathcal{P}(pk, u, w) : \mathcal{V}(vk, u, \pi) = 1\right] = 1.$$

In our constructions (see Chapter 4) we will get zero-knowledge even if the adversary knows the secret verification key, a strong type of zero-knowledge called composable zero-knowledge in [Gro06] due to it making composition of zero-knowledge proofs easier.

Definition 16 (Composable zero-knowledge)**.** $(\mathsf{RKGen}, \mathcal{P}, \mathcal{V})$ *is computationally compos-*

able zero-knowledge if for all probabilistic polynomial time stateful adversaries $\mathcal{A}$

$$\Pr\Big[(pk,vk) \leftarrow \mathsf{RKGen}(1^\lambda); (u,w) \leftarrow \mathcal{A}(pk,vk); \pi \leftarrow \mathcal{P}(pk,u,w) : (u,w) \in R_\lambda \text{ and } \mathcal{A}(\pi) = 1\Big]$$
$$\approx \Pr\Big[(pk,vk) \leftarrow \mathsf{RKGen}(1^\lambda); (u,w) \leftarrow \mathcal{A}(pk,vk); \pi \leftarrow \mathcal{S}(vk,u) : (u,w) \in R_\lambda \text{ and } \mathcal{A}(\pi) = 1\Big].$$

If the above holds also for unbounded stateful adversaries $\mathcal{A}$ then we say the argument is statistically composable zero-knowledge.

Culpable soundness [GOS12] is a relaxation of soundness that restricts the prover in the following way: First, we only consider false statements in a subset L_{guilt} of $\bar{L}_R$ characterised by a relation R_{guilt}. Second, we require a successful cheating prover to also output a guilt witness w_{guilt} along with his false statement u such that $(u, w_{\text{guilt}}) \in R_{\text{guilt}}$. Intuitively this definition captures the notion of a malicious prover being aware of the falsehood of the statement for which she is creating a fake proof.

Definition 17 (Adaptive culpable soundness)**.** *We say $(\mathcal{G},\mathcal{P},\mathcal{V})$ is culpably sound with respect to the relation R_{guilt} if for all probabilistic polynomial time $\mathcal{A}$*

$$\Pr\Big[(pk,vk) \leftarrow \mathcal{G}(1^\lambda); (u,\pi,w_{\text{guilt}}) \leftarrow \mathcal{A}(pk) : (u,w_{\text{guilt}}) \in R_{\text{guilt},\lambda} \text{ and } \mathcal{V}(vk,u,\pi) = 1\Big] \approx 0.$$

The above definition does not directly cover the adversary, $\mathcal{A}$ having access to a verification oracle $\mathcal{V}(vk,\cdot,\cdot)$. However it is straightforward to handle cases where the adversary has access to a logarithmic number of queries (as in [DFN06]), since that can be simulated by guessing the responses with inverse polynomial probability.

Chapter 4

Non-Interactive Zero Knowledge without Random Oracles

Damgård, Fazio and Nicolosi [DFN06] gave a transformation of Sigma-protocols, 3-move honest verifier zero-knowledge proofs, into efficient non-interactive zero-knowledge arguments for a designated verifier. Their transformation uses additively homomorphic encryption to encrypt the verifier's challenge, which the prover uses to compute an encrypted answer. The transformation does not rely on the random oracle model but proving soundness requires a complexity leveraging assumption.

We propose an alternative instantiation of original DFN transformation and show that it achieves culpable soundness without complexity leveraging. This improves upon an earlier result by Ventre and Visconti [VV09], who used a different construction which achieved *weak* culpable soundness, by achieving standard culpable soundness as well as avoiding the duplicate proofs required by their approach.

We demonstrate how our construction can be used to prove validity of encrypted votes in a referendum. This yields a voting system with homomorphic tallying that does not rely on the Fiat-Shamir heuristic.

The research presented in this chapter was joint work with Jens Groth and published in [CG15]. The author contributed to the investigation in defining strongly additively homomorphic encryption, the security proofs and the design and security proof of the referendum scheme. The idea of using culpable soundness and the boolean circuit application were put forward by Jens Groth.

4.1 Introduction

Cryptographic applications often require a party to demonstrate that a statement is true without revealing any additional details. For example, a voter may wish to prove that an encrypted message contains a vote for a valid candidate without disclosing the actual candidate. This can be done using *zero-knowledge* proofs [GMR89] that enable a prover to demonstrate to a verifier that a statement u belongs to a language L in NP defined by a relation R without giving the verifier any information about the witness w such that $(u,w) \in R$.

The Damgård, Fazio and Nicolosi (DFN) [DFN06] transformation works in the Registered Key Model (RKM) [BCNP04] where a verifier registers a public key and transforms a Σ-protocol with linear answer into a non-interactive zero-knowledge argument that can be verified by this specific verifier [JSI96]. The transformation works by having the verifier encrypt his challenge under an additively homomorphic encryption scheme and relies on the Σ-protocol having an answer that can be computed using linear algebra and the homomorphic property of the encryption scheme to enable the prover to complete an encrypted version of the answer in the Σ-protocol. We describe their construction in Section 4.3, where we give an alternate soundness proof for a more restricted class of protocols and encryption schemes. The original proof holds for a logarithmic number of proofs but soundness rests on a complexity leveraging assumption. Complexity leveraging assumptions state that there is a hardness separation between two hard problems, i.e. that there exist solvers for the first problem, running in time T, but no T-bound adversary can solve the second problem with more than negligible probability. This allows security reductions to raise a contradiction by solving the second, harder, problem using the adversary, and a solver for the first problem. Ventre and Visconti [VV09] give an alternative proof of soundness for a construction based on a two ciphertext variation of the DFN transformation in the style of Naor and Yung [NY90]. They replace the complexity leveraging assumption by introducing a modification of culpable soundness[1] [GOS12] that they call *weak culpable soundness*. Culpable soundness restricts adversaries to being "aware" of the falsehood of the statement they are proving. *Weak* culpable soundness furthermore requires that the adversary is also aware of the fact that she has succeeded in producing a convincing proof of a false statement, by producing a second auxiliary proof to that effect.

In the DFN setting using weak culpable soundness would require the adversary to

[1]Culpable soundness was also called co-soundness in an earlier version of [GOS12].

prove statements containing ciphertexts addressed to the designated verifier. It would be challenging to provide such an adversary with enough power to perform the required proofs without having knowledge of the verifier's secret decryption key. We instead opt to construct the underlying protocol with the property that forged proofs reveal the challenge. This is enough to contradict the semantic security of the encryption scheme used for the designated verifier proof if a false proof is ever produced.

4.1.1 Contribution

We give an instantiation of the DFN transformation that achieves standard culpable soundness without complexity leveraging. The transformation relies on an IND-CPA secure additively homomorphic encryption scheme and is quite efficient. The transformation can be applied to Σ-protocols that have linear answers and unique identifiable challenges (Sect. 4.2.3).

We can use our resulting non-interactive zero-knowledge designated verifier arguments to efficiently prove statements about encrypted plaintexts. In particular, we can prove that a ciphertext contains either 0 or 1 without disclosing the plaintext. This can in turn be used to prove that a set of ciphertexts encrypt a witness for the satisfiability of a circuit. For the appropriate Σ-protocols to be in place, we require the encryption scheme to be additively homomorphic modulo a prime and satisfy a few other requirements (Sect. 4.2.1). We use Okamoto-Uchiyama encryption [OU98] as an example.

We proceed to give an example application of our non-interactive zero-knowledge arguments to provide publicly verifiable arguments in the context of electronic voting. In voting systems such as Helios [Adi08] voters submit their votes encrypted under a homomorphic encryption scheme accompanied with non-interactive arguments (typically using the Fiat-Shamir transformation) that the encrypted votes are in fact valid. Ciphertexts with convincing arguments are aggregated homomorphically to produce an encrypted tally which is then decrypted to produce the result. By releasing the designated verifier keys to the public (similar to [Wik08]), once vote submission has concluded, we can use our non-interactive designated verifier arguments in place of the usual non-interactive zero-knowledge arguments with minimal changes to the design. It does however differ in that invalid ballots are detected after being tentatively accepted, whereas other systems are able to reject them outright.

Additionally, we require that the contents of the message board are initially hidden and only revealed after ballot submission has concluded. This is a departure from common practice, and raised the amount of trust placed on the bulletin board. However, it is not dissimilar from the way conventional elections are run, and the delay may be omitted by using a stronger (non-malleable) encryption.

4.2 Preliminaries

In this section we expand on some concepts from Chapter 3. This is necessary as the technical parts of this chapter rely on specific bit sizes of parameters and additional properties for our encryption scheme.

4.2.1 Strongly Additively Homomorphic Encryption

We say that $(\mathrm{PKEGen}, \mathcal{E}, \mathcal{D})$ is *additively homomorphic* if the randomness and ciphertext spaces are finite groups as well (written additively and multiplicatively respectively) and for all possible keys ek and plaintexts $m_1, m_2 \in \mathcal{M}_{ek}$ and $r_1, r_2 \in \mathcal{R}_{ek}$ we have

$$\mathcal{E}_{ek}(m_1; r_1) \cdot \mathcal{E}_{ek}(m_2; r_2) = \mathcal{E}_{ek}(m_1 + m_2; r_1 + r_2).$$

We say that an additively homomorphic scheme $(\mathrm{PKEGen}, \mathcal{E}, \mathcal{D})$ is a *strongly additively homomorphic scheme* if it satisfies the four additional properties described below:

Prime order message space: The message space is $\mathbb{Z}_p$ for some *prime* p.

Decryption homomorphic[2]**:** Membership of the ciphertext space can be efficiently tested and the decryption algorithm on all elements in $\mathcal{C}_{ek}$ returns a plaintext in $\mathcal{M}_{ek}$ (i.e., decryption does not fail). Furthermore, decryptions respect the additively homomorphic operation, i.e., for all possible key pairs (ek, dk) and $c_1, c_2 \in \mathcal{C}_{ek}$ we have

$$\mathcal{D}_{dk}(c_1) + \mathcal{D}_{dk}(c_2) = \mathcal{D}_{dk}(c_1 \cdot c_2).$$

Extended randomness: $R_{ek} = \mathbb{Z}_N$ for some integer N but the encryption function accepts randomness in $\mathbb{Z}$ and for all $m \in \mathcal{M}_{ek}$ and $r \in \mathbb{Z}$

$$\mathcal{E}(m; r) = \mathcal{E}(m; r \bmod N).$$

Verifiable keys: There exists an efficient test $\mathrm{VerifyKey}(1^\lambda, ek, dk)$ that given a public key ek and decryption key dk (or without loss of generality the randomness used in the

[2]This property is trivial for cryptosystems where the entire cipherspace consists of valid encryptions but in the general case it must be stated explicitly.

key generation) returns 1 if and only if (ek, dk) is a valid key pair using security parameter λ.

The strengthening of the usual homomorphic properties is mainly aimed to capitalize on our ability (via culpable soundness) to extract from adversarially generated ciphertexts. The prime order message space help by ensuring that the values we will extract will give unique solutions, and the decryption homomorphic property ensures that the adversary will not be able to present malformed ciphertexts (which might be hard to extract from) that homomorphically add up to a valid one. Extended randomness is present for generality in the event that the modulus of the randomness space, N is unknown.

For notational convenience, we let $c^{\vec{z}}$ be the vector $(c^{z_1}, \ldots, c^{z_n})$ given a ciphertext c and a vector of integers $\vec{z} = (z_1, \ldots, z_n)$. Given a vector $\vec{w}$ we also define $\vec{c} \leftarrow \mathcal{E}_{ek}(\vec{w})$ as the vector of ciphertexts given by $(\mathcal{E}_{ek}(w_1), \ldots, \mathcal{E}_{ek}(w_n))$.

4.2.2 Okamoto-Uchiyama encryption [OU98]

The Okamoto-Uchiyama [OU98] cryptosystem is strongly additively homomorphic with a message space $\mathbb{Z}_p$ for a prime p that is implicitly defined by the public key.

PKEGen(1^λ): Pick two different $\ell_p(\lambda)$-bit primes p, q and let $N = p^2 \cdot q$. Then choose a random g in $\mathbb{Z}_N^*$ such that $g \bmod p^2$ has order $p(p-1)$ in $\mathbb{Z}_{p^2}^*$. The public key is $ek = (N, g)$ and the secret decryption key is $dk = (ek, p)$.

$\mathcal{E}_{ek}(m)$: Given $m \in \mathbb{Z}_p$ return $\mathcal{E}_{ek}(m; r) = g^{m+rN} \bmod N$, where $r \leftarrow \mathbb{Z}_N$.

$\mathcal{D}_{dk}(c)$: Return $m = \frac{L(c^{p-1} \bmod p^2)}{L(g^{p-1} \bmod p^2)} \bmod p$, where $L(x) = \frac{x-1}{p}$.

For a given public key $ek = (N, g)$ the randomness space is $\mathbb{Z}_N$ and the ciphertext space is $\mathbb{Z}_N^*$. Even though the message space is defined as $\mathbb{Z}_p$, in practice we cannot disclose p but as long as the encrypting party picks messages $m \in \{0,1\}^{\ell_p(\lambda)-1}$ we are guaranteed that they fall within the message space and will decrypt correctly.

Direct calculation confirms that Okamoto-Uchiyama encryption is decryption homomorphic and that it is easy to extend the randomness space to $R_{ek} = \mathbb{Z}$. The keys are verifiable in the sense that given the decryption key, i.e., the factorization of N, it is easy to check that the keys are a valid output of the key generation algorithm and that the encryption scheme satisfies all the required properties.

4.2.3 Σ-protocols with Linear Answers and Unique Identifiable Challenges

We will restrict ourselves to Σ-protocols with a *linear answer over the integers*. By this we mean without loss of generality that we can consider a prover that generates the initial message a and two integer vectors $\vec{z}_1$ and $\vec{z}_2$. We consider protocols where x is picked as a random λ-bit string, where λ is the security parameter. The answer to a challenge $x \in \{0,1\}^\lambda$ can then be computed as the integer vector $\vec{z} = x\vec{z}_1 + \vec{z}_2$. We will assume that all the integers in $\vec{z}_1, \vec{z}_2, \vec{z}$ are non-negative and that there is a known polynomial upper bound $\ell_z(\lambda)$ on the bit-size of the integers.

We can describe a Σ-protocol for an NP-relation R with linear answer as a pair $(\mathcal{P}_\Sigma, \mathcal{V}_\Sigma)$, where $\mathcal{P}_\Sigma, \mathcal{V}_\Sigma$ are probabilistic polynomial time algorithms. The Σ-protocol runs as follows:

$(a, \vec{z}_1, \vec{z}_2) \leftarrow \mathcal{P}_\Sigma(x, w)$: The prover given a statement and witness pair $(u, w) \in R_\lambda$ generates an initial message a and a state $\vec{z}_1, \vec{z}_2$.

$x \leftarrow \{0,1\}^\lambda$: An λ-bit challenge is chosen uniformly at random.

$\vec{z} \leftarrow x\vec{z}_1 + \vec{z}_2$: An answer to the challenge x can be computed as $\vec{z} = x\vec{z}_1 + \vec{z}_2$.

$\{0,1\} \leftarrow \mathcal{V}_\Sigma(u, a, x, \vec{z})$: The verifier given a statement u and a protocol transcript $(a, x, \vec{z})$ returns 1 if accepting and 0 if rejecting. The verifier will always reject if any inputs are malformed, for instance if $x \notin \{0,1\}^\lambda$ or $\vec{z}$ contains an entry $z_i \notin \{0,1\}^{\ell_z(\lambda)}$.

With respect to soundness, we will for our purposes be interested in a special class of Σ-protocols that have unique identifiable challenges. Traditionally, Σ-protocols are required to have *2-special soundness*, which says that if the prover, after having created the initial message a, can answer two different challenges x and x' then it is possible to compute a witness w for the statement u being proved such that $(u, w) \in R$.

We do not need the witness to be extractable, and will therefore relax the soundness definition to just saying that on a false statement there is at most a single unique challenge the prover can answer after having created the initial message a.

However, we will require that under certain circumstances this unique answerable challenge should be identifiable, i.e., if the prover "knows" the statement is false in a certain way then she can actually compute the unique challenge x she will be able to answer if she can answer any challenge at all. We define this by adapting the notion of culpable soundness from [GOS12] and strong soundness from [HN06]. We say that the unique challenge is identifiable using an NP-relation R_{guilt}, which only contains false statements, if when the prover produces a statement x and a witness w_{guilt} of being guilty of cheating such that $(u, w_{\text{guilt}}) \in R_{\text{guilt}}$, then it is possible to efficiently compute a unique challenge where the verifier may possibly accept (in comparison, strong soundness omits R_{guilt}). The relation R_{guilt} will typically include all false statements that have a special form, depending on the specifics.

Definition 18 (Soundness with unique identifiable challenge)**.** *We say $(\mathcal{P}_\Sigma, \mathcal{V}_\Sigma)$ has a unique identifiable challenge using NP-relation R_{guilt} if there is a polynomial time algorithm E that takes as input the statement, witness and initial message and returns the unique challenge x that can be answered. Formally, we require that for all $\lambda, u, w_{\text{guilt}}, a, x, z$ where $(u, w_{\text{guilt}}) \in R_{\text{guilt},\lambda}$ and $\mathcal{V}_\Sigma(u, a, x, z) = 1$ that $x = E(u, w_{\text{guilt}}, a)$.*

A frequently asked question is why would the adversary want to provide a witness for cheating. The answer is that there are many natural scenarios where the real adversary is only a part of a larger system that contains the guilt witness. It may well be that the system would never provide a guilt witness in a normal execution but even when that is the case the notion can still be useful in security proofs: by framing a “standard” adversary within such a system we are able to explicitly use privileged information held by honest parties in security reductions. In Sect. 4.4 we give voting as a concrete example of how culpable soundness can be used to prevent cheating by voters. Voters prove that they have encrypted valid votes using the election system’s public key. The guilt witness is the decryption key, which the voting system will never make public since it would reveal all the votes. However, if a cheating voter exists, it is enough to point out that the guilt witness will exist in the possession of the electoral authorities. To satisfy the definition we may consider a new adversary which consists of the cheating voter’s behaviour, with the decryption key added to the output in a post-processing step. Culpable soundness then guarantees the voter cannot cheat and submit an invalid vote.

We note that the extractor E only requires the guilt witness and the initial message from the prover. This will be critical in the next section where the protocol is made non-interactive via the DFN transformation and the prover’s answer will be encrypted. In general, we cannot require that a cheating prover knows the contents of that ciphertext since it might have been assembled in a way that differs from the protocol.

4.2.4 Σ-protocol for additively homomorphic encryption of 0 or 1.

Consider a strongly additively homomorphic encryption scheme $(\mathcal{G},\mathcal{E},\mathcal{D})$ with message space $\mathbb{Z}_p$ for a prime p defined by the encryption key. We will now give a Σ-protocol for proving that a ciphertext encrypts 0 or 1 using randomness $r \in \{0,1\}^{\ell_r(\lambda)}$ bounded by a polynomial $\ell_r(\lambda)$.

Prover$((ek,c),(m,r))$		Verifier(ek,c)
$m_a \leftarrow \{1\}\|\|\{0,1\}^{2\lambda}$	$\xrightarrow{a,b}$	Accept if and only if
$r_a \leftarrow \{0,1\}^{\ell_r(\lambda)+2\lambda}; a \leftarrow \mathcal{E}_{ek}(m_a;r_a)$		$a,b,c \in \mathcal{C}_{ek}, f \in \{0,1\}^{2\lambda+2}$
$r_b \leftarrow \{0,1\}^{\ell_r(\lambda)+3\lambda}; b \leftarrow \mathcal{E}_{ek}(-mm_a;r_b)$	$\xleftarrow{x \leftarrow \{0,1\}^{\lambda}}$	$z_a \in \{0,1\}^{\ell_r(\lambda)+2\lambda+1}$
		$z_b \in \{0,1\}^{\ell_r(\lambda)+3\lambda+1}$
$f := xm+m_a, z_a := xr+r_a$	$\xrightarrow{f,z_a,z_b}$	$c^x a = \mathcal{E}_{ek}(f;z_a)$
$z_b := (f-x)r+r_b$		$c^{f-x}b = \mathcal{E}_{ek}(0;z_b)$

Figure 4.1: Σ-protocol for encryption of 0 or 1.

Let

$$R = \Big\{\Big((ek,c),(m,r)\Big) : m \in \{0,1\} \text{ and } r \in \{0,1\}^{\ell_r(n)} \text{ and } c = \mathcal{E}_{ek}(m;r)\Big\},$$

$$R_{\text{guilt}} = \Big\{\Big((ek,c),dk\Big) : c \in \mathcal{C}_{ek} \text{ and } \mathcal{D}_{dk}(c) \notin \{0,1\} \text{ and VerifyKey}(1^{\lambda},ek,dk) = 1\Big\}.$$

Theorem 1. *Fig. 4.1 describes a Σ-protocol for R with linear answer and unique identifiable challenge using R_{guilt} assuming $(\mathcal{G},\mathcal{E},\mathcal{D})$ is a strongly additively homomorphic encryption scheme with message space $\mathbb{Z}_p$ of sufficiently large size such that $\ell_p(\lambda) > \lambda$.*

Proof. The algorithms are probabilistic polynomial time. The protocol has linear answer with a polynomial upper bound of $\ell_z(\lambda) = \ell_r(\lambda) + 3\lambda + 1$ on the bit-lengths of the integers in the answer. Direct verification shows that the protocol is perfectly complete.

The protocol is statistical SHVZK. The simulator, given challenge $x \in \{0,1\}^{\lambda}$ picks $f \leftarrow \{1\}\|\{0,1\}^{2\lambda}$, $z_a \leftarrow \{0,1\}^{\ell_r(\lambda)+2\lambda}$ and $z_b \leftarrow \{0,1\}^{\ell_r(\lambda)+3\lambda}$. It then computes $a = c^{-x}\mathcal{E}_{ek}(f;z_a)$ and $b = c^{x-f}\mathcal{E}_{ek}(0;z_b)$ and returns the simulated proof (a,b,f,z_a,z_b). Observe that the simulated f,z_a,z_b are statistically close to those of a real proof. To see the simulation is statistically indistinguishable from a real proof with challenge x all that remains to be seen is that given f,z_a,z_b, the initial message containing a,b is fixed by the verification equations in both real and simulated proofs.

Finally, let us show that the protocol has unique identifiable challenges using R_{guilt}. A witness in R_{guilt} gives us the decryption key for the encryption scheme. We can verify the correctness of the decryption key and decrypt c to get m and also decrypt a, b to get plaintexts m_a and m_b. In a succesful argument, the value f must be $f = xm + m_a \bmod p$ since otherwise the first verification equation would fail. The second verification equation gives us $(f - x)m + m_b = 0 \bmod p$, which means $e(m-1)m + m_a m + m_b = 0 \bmod p$. If $m \notin \{0,1\}$ we have that $(m-1)m \neq 0 \bmod p$ and therefore the equation uniquely determines $x \bmod p$. With $p > 2^{\lambda}$ this identifies a unique challenge $x \in \{0,1\}^{\lambda}$ that the prover may be able to answer or shows that no answerable challenge exists. $\square$

4.3 Transformation

We will now use the DFN transformation [DFN06] on a Σ-protocol with linear answer over the integers and unique identifiable challenges to get a non-interactive designated verifier argument. The transformation we use is identical to [DFN06], but the soundness proof differs: at the cost of requiring soundness with unique identifiable challenge, a strongly additively homomorphic encryption scheme and proving culpable soundness, we eschew the use of complexity leveraging assumptions.

The verifier uses an additively homomorphic encryption scheme $(\mathsf{RKGen}, \mathcal{E}, \mathcal{D})$ to encrypt a random challenge x. Since the Σ-protocol has linear answer, the prover can now use the homomorphic property of the encryption scheme to compute an encryption of the answer $\vec{z}$ in the Σ-protocol, which is sent together with the initial message a. The verifier decrypts the ciphertext from the prover to get z and checks whether $(a, x, \vec{z})$ is a valid proof. The full non-interactive designated verifier argument is described in Fig. 4.2.

$\mathsf{RKGen}(1^{\lambda})$	$\mathcal{P}(pk, u, w)$	$\mathcal{V}(vk, u, \pi)$
$(ek, dk) \leftarrow \mathsf{PKEGen}(1^{\lambda})$	$(a, \vec{z}_1, \vec{z}_2) \leftarrow \mathcal{P}_{\Sigma}(u, w)$	Parse $\pi = (a, \vec{c}_z)$
$x \leftarrow \{0,1\}^{\lambda}$	$\vec{c}_z \leftarrow c^{\vec{z}_1}\mathcal{E}_{ek}(\vec{z}_2)$	$\vec{z} \leftarrow \mathcal{D}_{dk}(\vec{c}_z)$
$c \leftarrow \mathcal{E}_{ek}(x)$	Return $\pi := (a, \vec{c}_z)$	Return $\mathcal{V}_{\Sigma}(u, a, x, \vec{z})$
$pk := (ek, c)$		
$vk := (dk, x)$		
Return (pk, vk)		

Figure 4.2: Non-interactive designated verifier argument

Theorem 2. $(\mathsf{RKGen}, \mathcal{P}, \mathcal{V})$ *specified in Fig. 4.2 is a non-interactive designated verifier argument for R with culpable soundness for R_{guilt} if $(\mathcal{P}_{\Sigma}, \mathcal{V}_{\Sigma})$ is a Σ-protocol for R with linear* *answer over the integers and soundness with unique identifiable challenge using R_{guilt} and if $(\text{PKEGen}, \mathcal{E}, \mathcal{D})$ is an additively homomorphic, IND-CPA secure public key encryption scheme where $\mathbb{Z}_p$ is of sufficiently large size to include the answers, i.e., $\ell_p(\lambda) > \ell_z(\lambda)$.*

Proof. Since $(\mathcal{P}_\Sigma, \mathcal{V}_\Sigma)$ and $(\text{PKEGen}, \mathcal{E}, \mathcal{D})$ are probabilistic polynomial time algorithms so are $(\mathsf{RKGen}, \mathcal{P}, \mathcal{V})$. Perfect completeness follows from the additive homomorphicity of the encryption scheme and that $0 \leq z_i < 2^{\ell_z(\lambda)} < p$ for all entries z_i in $\vec{z}$ combined with the perfect completeness of $(\mathcal{P}_\Sigma, \mathcal{V}_\Sigma)$.

Next, we will prove that the construction is zero-knowledge. The simulator knows the secret verification key $vk = (dk, x)$. It starts by running the SHVZK simulator for the Σ-protocol to get a simulated proof $(a, x, \vec{z})$ for the statement u. It then generates $\vec{c}_z \leftarrow \mathcal{E}_{ek}(\vec{z})$ and returns the simulated argument $\pi := (a, \vec{c}_z)$.

To see a that simulated argument is indistinguishable from a real argument consider a hybrid simulator that does get the witness as input. This hybrid simulator proceeds by following the Σ-protocol to get an argument $(a, x, \vec{z})$ and then encrypts $\vec{z}$ to get $\vec{c}_z$. Since the encryption scheme is also homomorphic with respect to the randomness used for encryption, the hybrid arguments generated this way and real arguments are perfectly indistinguishable. Furthermore, since the Σ-protocol is SHVZK, hybrid arguments and simulated arguments are computationally indistinguishable. Furthermore, if the Σ-protocol has statistical SHVZK then the hybrid arguments and simulated arguments are statistically indistinguishable.

Finally, we will prove that the construction has adaptive culpable soundness with respect to R_{guilt}. Plugging our construction into the probability defining culpable soundness with a probabilistic polynomial time adversary $\mathcal{A}$ we get

$$\Pr\left[\begin{array}{ll} (ek, dk) \leftarrow \mathsf{RKGen}(1^\lambda); x \leftarrow \{0,1\}^\lambda; c \leftarrow \mathcal{E}_{ek}(x) & (u, w_{\text{guilt}}) \in R_{\text{guilt}} \\ (u, (a, \vec{c}_z), w_{\text{guilt}}) \leftarrow \mathcal{A}(ek, c); \vec{z} \leftarrow \mathcal{D}_{dk}(\vec{c}_z) & \mathcal{V}_\Sigma(u, a, x, \vec{z}) = 1 \end{array} : \right].$$

By the unique identifiable challenge property of the Σ-protocol this probability is at most the chance that x is the unique answerable challenge:

$$\Pr\left[\begin{array}{ll} (ek, dk) \leftarrow \mathsf{RKGen}(1^\lambda); x \leftarrow \{0,1\}^\lambda; c \leftarrow \mathcal{E}_{ek}(x) & (u, w_{\text{guilt}}) \in R_{\text{guilt}} \\ (u, (a, \vec{c}_z), w_{\text{guilt}}) \leftarrow \mathcal{A}(ek, c); \vec{z} \leftarrow \mathcal{D}_{dk}(\vec{c}_z) & x = E(u, w_{\text{guilt}}, a) \end{array} : \right].$$

By the IND-CPA security of the encryption scheme, this probability is at most negligibly larger than the same expression with c encrypting a random challenge x'

$$\Pr\left[\begin{array}{l}(ek,dk)\leftarrow \mathsf{RKGen}(1^\lambda);x,x'\leftarrow\{0,1\}^\lambda;c\leftarrow\mathcal{E}_{ek}(x') \\ (u,(a,\vec{c}_z),w_{\text{guilt}})\leftarrow\mathcal{A}(ek,c);\vec{z}\leftarrow\mathcal{D}_{dk}(\vec{c}_z)\end{array} : \begin{array}{c}(u,w_{\text{guilt}})\in R_{\text{guilt}} \\ x=E(u,w_{\text{guilt}},a)\end{array}\right].$$

Since x is chosen uniformly random this latter probability is at most $2^{-\lambda}$, which is negligible. □

4.3.1 Non-interactive Designated Verifier Arguments for Statements about Ciphertexts

In Sect. 4.2.4 we gave a Σ-protocol for proving a ciphertext having either 0 or 1 as plaintext. Using the DFN transformation, this leads to a non-interactive designated verifier argument with culpable soundness for a ciphertext encrypting 0 or 1, i.e., for the relation

$$R=\left\{((ek,c),(m,r)):m\in\{0,1\}\text{ and }r\in\{0,1\}^{\ell_r(n)}\text{ and }c=\mathcal{E}_{ek}(m;r)\right\}$$

with culpable soundness using

$$R_{\text{guilt}}=\left\{((ek,c),dk):c\in\mathcal{C}_{ek}\text{ and }\mathcal{D}_{dk}(c)\notin\{0,1\}\text{ and VerifyKey}(1^\lambda,ek,dk)=1\right\}.$$

This designated verifier argument works for ciphertexts produced by all strongly additively homomorphic encryption schemes that have message space $\mathbb{Z}_p$ for $p>2^\lambda$ such as for instance the Okamoto-Uchiyama [OU98] encryption scheme from Sect. 4.2.2. A second instance of the same strongly additively homomorphic encryption scheme but with larger message space can also be used for the DFN transformation. However, in the interest of more efficient implementations, it might be desirable to use a different encryption scheme for the DFN transformation. Specifically, DFN does not require the message space to be of prime order or the scheme to be *strongly* additively homomorphic, giving us the option of using an encryption scheme better suited for encrypting long messages such as Damgård-Jurik [DJN10].

It is fairly simple to adapt standard Σ-protocols for other languages expressing properties about ciphertexts. In particular, in addition to the argument for encryption of 0 or 1 it is possible to construct non-interactive designated verifier arguments for the following relations:

Plaintext is 0: We can prove that a ciphertext c encrypts 0, i.e., give a non-interactive designated verifier argument for the relation

$$R^0 = \Big\{((ek,c),r) : r \in \{0,1\}^{\ell_r(n)} \text{ and } c = \mathcal{E}_{ek}(0;r)\Big\}.$$

Equivalence of plaintexts: Given two ciphertexts c and c', we can give a non-interactive designated verifier argument for them having the same plaintext by proving that c/c' is an encryption of 0 using the above designated verifier argument.

Multiplicative relationship: Given a triple of ciphertexts c_0, c_1 and c_2, we can prove that the plaintexts m_0, m_1 and m_2 satisfy $m_0 = m_1 m_2 \bmod p$. More precisely, we can construct a designated verifier argument for the relation

$$R^M = \left\{ \begin{array}{r} ((ek,c_0,c_1,c_2),(m_1,m_2,r_0,r_1,r_2)) : m_1,m_2 \in \mathbb{Z}_p, r_0,r_1,r_2 \in \{0,1\}^{\ell_r(n)} \\ c_0 = \mathcal{E}_{ek}(m_1m_2;r_0) \text{ and } c_1 = \mathcal{E}_{ek}(m_1;r_1) \text{ and } c_2 = \mathcal{E}_{ek}(m_2;r_2) \end{array} \right\}.$$

In all cases, the corresponding guilt witness w_{guilt} consists of the decryption key, which can be used to decrypt the ciphertexts in the statement.

4.3.2 Circuit Satisfiability

We will now show that given a circuit consisting of NAND-gates and encryptions of the wires it is possible to prove that the plaintexts correspond to a satisfying assignment. A circuit C with $k+1$ wires and s gates can be described as $\{(j_1, j_2, j_3)\}_{j=1}^s$, which means that the wires should satisfy $w_{j_3} = \neg(w_{j_1} \wedge w_{j_2})$. We let the output wire be $w_0 = 1$ and the corresponding ciphertext be $c_0 = \mathcal{E}_{ek}(1;0)$ encrypted with randomness $r_0 = 0$. We consider the relations:

$$R^C = \left\{ \begin{array}{r} ((C,ek,c_1,\ldots,c_k),(w_1,r_1,\ldots,w_k,r_k)) \mid \forall j = 1,\ldots,s : w_{j_3} = \neg(w_{j_1} \wedge w_{j_2}) \\ \forall i = 1,\ldots,k : w_i \in \{0,1\} \wedge r_i \in \{0,1\}^{\ell_r(\lambda)-2} \wedge c_i = \mathcal{E}_{ek}(w_i;r_i) \end{array} \right\},$$

$$R^C_{\text{guilt}} = \left\{ \begin{array}{l} ((C,ek,c_1,\ldots,c_k),dk) \mid \text{VerifyKey}(1^\lambda,ek,dk) = 1 \text{ and } \forall i = 1,\ldots k : c_i \in \mathcal{C}_{ek} \\ \exists i \in \{1,\ldots,k\} : w_i = \mathcal{D}_{dk}(c_i) \notin \{0,1\} \text{ or } \exists j \in \{1,\ldots,s\} : w_{j_3} \neq \neg(w_{j_1} \wedge w_{j_2}) \end{array} \right\}.$$

The strategy in the designated verifier argument for R^C is to first prove that each ciphertext contains a wire value $w_i \in \{0,1\}$. Next, the prover proves for each NAND-gate

(j_1, j_2, j_3) that $w_{j_3} = \neg(w_{j_1} \wedge w_{j_2})$. Following [GOS12] we have for $w_{j_1}, w_{j_2}, w_{j_3} \in \{0,1\}$

$$w_{j_3} = \neg(w_{j_1} \wedge w_{j_2}) \quad \text{if and only if} \quad w_{j_1} + w_{j_2} + 2w_{j_3} - 2 \in \{0,1\}.$$

Using the homomorphic properties of the encryption scheme, we will therefore for each NAND-gate show $c_{j_1} c_{j_2} c_{j_3}^2 \mathcal{E}_{ek}(-2;0)$ contains 0 or 1. The full construction can be found in Fig. 4.3

```
$\mathcal{P}^C(pk, (C, ek, c_1, \ldots, c_k), (w_1, r_1, \ldots, w_k, r_k))$
$w_0 = 1, r_0 = 0, c_0 = \mathcal{E}_{ek}(w_0; r_0)$
For $i = 1, \ldots, k$
    $\pi_i \leftarrow \mathcal{P}(pk, c_i, (w_i, r_i))$
Parse $C = \{(j_1, j_2, j_3)\}_{j=1}^{s}$
For $j = 1, \ldots, s$
    $c'_j = c_{j_1} c_{j_2} c_{j_3}^2 \mathcal{E}_{ek}(-2;0)$
    $m'_j = w_{j_1} + w_{j_2} + 2w_{j_3} - 2$
    $r'_j = r_{j_1} + r_{j_2} + 2r_{j_3}$
    $\pi'_j \leftarrow \mathcal{P}(pk, c'_j, (m'_j, r'_j))$
Return $\pi = (\pi_1, \ldots, \pi_k, \pi'_1, \ldots, \pi'_s)$
```

```
$\mathcal{V}^C(vk, (C, ek, c_1, \ldots, c_k), \pi)$
$w_0 = 1, r_0 = 0, c_0 = \mathcal{E}_{ek}(w_0; r_0)$
Parse $C = \{(j_1, j_2, j_3)\}_{j=1}^{s}$
For $j = 1, \ldots, s$
    $c'_j = c_{j_1} c_{j_2} c_{j_3}^2 \mathcal{E}_{ek}(-2;0)$
Parse $\pi = (\pi_1, \ldots, \pi_k, \pi'_1, \ldots, \pi'_s)$
Accept if and only if
    For $i = 1, \ldots, k$
        $\mathcal{V}(vk, c_i, \pi_i) = 1$
    For $j = 1, \ldots, s$
        $\mathcal{V}(vk, c'_j, \pi'_j) = 1$
```

Figure 4.3: Non-interactive designated verifier argument $(G^C, \mathcal{P}^C, \mathcal{V}^C)$ for encryption of satisfying assignment of wires in a circuit using $\mathcal{G}^C = \mathcal{G}$ where $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is a designated verifier argument for encryption of 0 or 1.

Theorem 3. *$(\mathcal{G}^C, \mathcal{P}^C, \mathcal{V}^C)$ given in Fig. 4.3 is a non-interactive designated verifier argument for R^C with culpable soundness using R^C_{guilt} if $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is a non-interactive designated verifier argument for encryption of 0 or 1 using R_{guilt} from Sect. 4.2.4.*

Proof. Perfect completeness follows from the homomorphic properties of the encryption scheme and the perfect completeness of $(\mathcal{G}, \mathcal{P}, \mathcal{V})$.

We will now prove composable zero-knowledge. The simulator $\mathcal{S}^C(vk, (C, ek, c_1, \ldots, c_k))$ runs like the prover except it simulates the proofs $\pi_1, \ldots, \pi_k, \pi'_1, \ldots, \pi'_s$ as $\pi_i \leftarrow \mathcal{S}(vk, (ek, c_i))$ and $\pi'_j \leftarrow \mathcal{S}(vk, c'_j)$. We prove this via a hybrid argument. Let $SP_i(vk, (C, ek, c_1, \ldots, c_k), (w_1, r_1, \ldots, w_k, r_k))$ for $i = 0 \ldots (k+s)$ by a hybrid prover/simulator, operating as follows: it uses the real prover for the first $(k+s-i)$ subproofs of π, and the simulator for the last i. It is clear that SP_0 behaves as the real prover (since it ignores its first argument), while $SP_{(k+s)}$ behaves as the simulator (since it ignores its last argument). Thus, if an adversary $\mathcal{A}^C$ can distinguish between the prover (i.e. SP_0) and simulator ($SP_{(k+s)}$), there must also be a value i so that is distinguishes SP_i from $SP_{(i+1)}$. We will use $\mathcal{A}^C$ to build an adversary $\mathcal{A}$ against the composable zero knowledge of $(\mathcal{G},\mathcal{P},\mathcal{V})$. $\mathcal{A}$ operates by running $\mathcal{A}^C$ to obtain a statement for the circuit protocol, and isolates the ciphertext c and witness r used in subproof $i+1$. It then returns c,r to the game, obtaining π^*, finally it creates a proof π by running SP_i, but inserting π^* in place of subproof $i+1$, and copies the response of $\mathcal{A}^C$. It is clear that iff $\mathcal{A}^C$ correctly guesses whether π was produced identically to SP_i or $SP_(i+1)$ then $\mathcal{A}$ correctly guesses if π^* was created by the prover or the simulator.

We remark that here the usefulness of *composable* zero-knowledge comes into play since it gives the adversary access to the verification key vk, which allows the hybrid argument to go through.

Finally, we will prove that the argument is culpably sound. By the culpable soundness of $(\mathsf{RKGen},\mathcal{P},\mathcal{V})$ when we are given dk by the adversary, the proofs $\pi_1,\dots,\pi_k$ guarantee each ciphertext contains 0 or 1. The homomorphic property of the encryption scheme combined with the culpable soundness of the proofs $\pi'_1,\dots,\pi'_s$ then shows that the plaintexts respect the NAND-gates. Since the output is $w_0 = 1$ this means the circuit is satisfied by the encrypted values. $\square$

4.4 Applications in Voting with Homomorphic Tallying

We will use a basic referendum voting scheme as an illustration of how to use non-interactive zero-knowledge designated verifier arguments with culpable soundness. We use a modification of the framework by Bernhard et al. [BCP$^+$11] which generalises the Helios voting system. Such schemes operate by having eligible voters post their votes on a bulletin board encrypted with an additively homomorphic encryption scheme. The election result can then be produced by a single decryption operation on the homomorphic sum of the individual votes. Zero-knowledge protocols ensure that the various participating parties remain honest.

4.4.1 Voting Schemes with Delayed Bulletin Boards

We assume a *delayed* bulletin board BB holds all messages posted by the various participants in the election, and that it behaves honestly for the entirety of the election. During the submission of ballots, it operates in an append-only mode *without* disclosing its contents. After voting has concluded, the bulletin board reveals the ballots it contains and checks their validity. The checks use only public information and as such are reproducible by any party;

the bulletin board performs them for convenience. Finally, we assume that the history of the bulletin board is publicly accessible as well as the current state.

Our use of a *delayed* bulletin board is a departure from usual practice and is aimed at preventing attacks based on malleability. The additional trust placed on the board by this requirement may be mitigated by having the bulletin board immediately display commitments to ballots or eliminated by augmenting the ballot encryption to be submission secure [Wik08]. We also note that Cortier et al. [CGGI13] develop techniques to guard against misbehaving boards.

In the interest of simplicity, we restrict the options in the referendum to $\{0,1\}$ without giving the option of casting an abstention ballot. The election is run by two trustees, $\mathcal{T}_D$ and T_V, tasked with holding the decryption and verification keys for the election. We will for simplicity consider them to be trusted parties but they could be implemented using threshold cryptography.

Definition 19 (Delayed Voting Scheme)**.** *A voting scheme* Π *consists of five probabilistic polynomial time algorithms:* **Setup***,* **Vote***,* **SubmitBallot***,* **CheckBoard***,* **Tally***, which operate as follows:*

Setup *The setup algorithm takes as input a security parameter* 1^λ*. It produces secret information SEC, public information PUB and verification information AUG. It also initialises the bulletin board BB and sets it to be hidden. PUB is assumed to be public knowledge after* **Setup** *has run.*

Vote *Vote accepts a vote* $m \in \{0,1\}$ *and outputs a ballot B encoding m.*

SubmitBallot *SubmitBallot*(B,BB) *takes as input a ballot B and the current state of the bulletin board BB and outputs either* $(0,BB)$ *if it rejects B or* $(1,BB\overset{+}{\leftarrow}B)$ *if it accepts it.*

CheckBoard *CheckBoard*(BB,AUG) *makes BB visible, and then checks all ballots on BB, replacing with* $\perp$ *any ballots that do not pass the verification tests. After checking, the verification information of valid ballots can be removed from the board.*

Tally *Tally* (BB,SEC) *takes as input a verified bulletin board BB and the secret information SEC and outputs the election result.*

For correctness we require that the ballots of honest voters are counted correctly, and that ballots cast by malicious voters cannot influence the election more than an honest one (i.e casting q malicious ballots can only add q votes and subtract none).

```
$\mathbf{Exp}^{COR}_{\Pi,\mathcal{A}}(\lambda)$
$(PUB, SEC, AUG) \leftarrow \mathbf{Setup}(1^\lambda)$
$(vsum, q) := (0, 0)$
$\mathcal{A}^{\mathbf{VoteOracle}(\cdot),\mathbf{BallotOracle}(\cdot)}(PUB)$
$BB \leftarrow \mathbf{CheckBoard}(BB, AUG)$
$result \leftarrow \mathbf{Tally}(BB, SEC)$
Return $(result, vsum, q)$
```

```
$\mathbf{VoteOracle}(\mathrm{v})$
$B \leftarrow \mathbf{Vote}(v)$
$(r, BB) \leftarrow \mathbf{SubmitBallot}(B, BB)$
if $r = accept$ : $vsum := (vsum + v)$
Return $(r, B)$

$\mathbf{BallotOracle}(\mathrm{B})$
$(r, BB) \leftarrow \mathbf{SubmitBallot}(B, BB)$
$q := q + 1$
Return $r$
```

Figure 4.4: The referendum correctness experiment, and the oracles provided to the adversary.

Definition 20 (Correctness)**.** *We say that a referendum voting scheme* Π *is correct if for all efficient adversaries* $\mathcal{A}$*:*

$$\Pr\left[(result, vsum, q) \leftarrow \mathbf{Exp}^{COR}_{\Pi,\mathcal{A}}(\lambda) : vsum \le result \le q + vsum\right] \approx 1$$

Definition 21 (Delayed Ballot Privacy)**.** *We say that a delayed voting scheme* Π *satisfies ballot privacy if for all efficient stateful interactive adversaries* $\mathcal{A}$*:*

$$\Pr\left[\mathbf{Exp}^{BP}_{\Pi,\mathcal{A}}(\lambda) = 1\right] \approx \frac{1}{2}$$

```
$\mathbf{Exp}^{BP}_{\Pi,\mathcal{A}}(\lambda)$
$(PUB, SEC, AUG) \leftarrow \mathbf{Setup}(1^\lambda)$
$b \leftarrow \{0, 1\}$
$\mathcal{A}^{\mathbf{VoteOracle}(\cdot),\mathbf{BallotOracle}(\cdot)}(PUB)$
$BB \leftarrow \mathbf{CheckBoard}(BB, AUG)$
$BB' \leftarrow \mathbf{CheckBoard}(BB', AUG)$
$result \leftarrow \mathbf{Tally}(BB', SEC)$
$\hat{b} \leftarrow \mathcal{A}(result, BB, AUG)$
Return $b = \hat{b}$
```

```
$\mathbf{VoteOracle}(\mathrm{v})$
$B' \leftarrow \mathbf{Vote}(v)$
if $b = 1$ then $B \leftarrow B'$
  else $B \leftarrow \mathbf{Vote}(0)$
$(r, BB) \leftarrow \mathbf{SubmitBallot}(B, BB)$
$(r', BB') \leftarrow \mathbf{SubmitBallot}(B', BB')$
Return $(r, B)$

$\mathbf{BallotOracle}(\mathrm{B})$
$(r, BB) \leftarrow \mathbf{SubmitBallot}(B, BB)$
if $r = accept$ then
  $(r', BB') \leftarrow \mathbf{SubmitBallot}(B, BB')$
Return $r$
```

Figure 4.5: The Ballot Privacy experiment, and the oracles provided to the adversary

4.4.2 A Referendum Voting Scheme

We will now describe a delayed voting scheme Π^{REF} for a yes-no referendum, based on an additively homomorphic encryption scheme such as $(\text{PKEGen}, \mathcal{E}, \mathcal{D})$ and with a non-interactive designated verifier argument system $(\mathsf{RKGen}, \mathcal{P}, \mathcal{V})$ for a plaintext being 0 or 1 such as the one given in Fig. 4.2.

For simplicity, we omit correctness proofs for keys having been generated correctly but point out that since the setup involves a limited number of parties we could assume the use of online zero-knowledge protocols using standard techniques. We also assume that the bulletin board behaves honestly. We now give descriptions of the Voting Protocol Algorithms:

Setup The setup algorithm takes as input a security parameter 1^{λ}. The decryption trustee $\mathcal{T}_D$ runs $\text{PKEGen}(1^{\lambda})$ to produce (ek, dk) and the verification trustee then runs $\mathsf{RKGen}(1^{l_z(\lambda)})$ to obtain (pk, vk). Let, $PUB = (ek, pk)$, $AUG = vk$ and $SEC = dk$. The procedure also initialises the bulletin board BB to be hidden, and publishes PUB.

Vote(m) Pick $r \leftarrow \{0,1\}^{\ell_r(\lambda)}$ and return (c, π), where $c = \mathcal{E}_{ek}(m; r)$ and $\pi \leftarrow \mathcal{P}(pk, (ek, c, r))$.

SubmitBallot(B,BB) Return $(accept, BB \overset{+}{\leftarrow} B)$.

CheckBoard(BB,AUG) The bulletin board BB becomes visible. $\mathcal{T}_V$ publishes AUG. For every ballot $B = (c, \pi)$ in BB we check whether $c \in \mathcal{C}_{ek}$ and $\mathcal{V}(vk, (ek, c), \pi) = 1$. If not, they will be omitted from the tally.

Tally(BB,SEC) The decryption trustee publishes $result = \mathcal{D}_{dk}(\prod_{i=1}^{k} c_i)$, where $c_1, \ldots, c_k$ are the encrypted votes that passed the validity check.

Theorem 4. *The referendum scheme Π^{REF} defined above is correct, if $(\mathsf{RKGen}, \mathcal{P}, \mathcal{V})$ is adaptive culpablby sound.*

Proof. Let $\mathcal{A}$ be an adversary against $\mathbf{Exp}^{COR}_{\Pi,\mathcal{A}}(\lambda)$ that causes *result* to be out of bounds with non-negligible probability. We construct a simulator $\mathcal{B}$ that contradicts the adaptive culpable soundness of $(\mathsf{RKGen}, \mathcal{P}, \mathcal{V})$. $\mathcal{B}$ will simulate the correctness experiment for $\mathcal{A}$ while acting as the adversary for the adaptive culpable soundness experiment. $\mathcal{B}$ operates by running the correctness experiment normally with the difference that it does not generate (pk, vk) but instead obtains pk from the adaptive culpable soundness experiment.

Because (PKEGen, $\mathcal{E}, \mathcal{D}$) is correct and additively homomorphic, *result* being out of bounds implies one of the submitted ballots $B = (c, \pi)$ is such that c encrypts a value other than 0 or 1 while at the same time $\mathcal{V}(vk, (ek, c), \pi) = 1$. Choosing one of the q ballots at random, $\mathcal{B}$ outputs $(u, \pi, w_{\text{guilt}})$ to the experiment, where $u = (c, pk)$ and $w_{\text{guilt}} = dk$. Since q is polynomial in λ this gives $\mathcal{B}$ a non-negligible probability of winning the experiment. $\square$

Theorem 5. *The scheme Π^{REF} satisfies delayed ballot privacy, if the* $(\mathsf{RKGen}, \mathcal{P}, \mathcal{V})$ *argument has statistical zero-knowledge, and* (PKEGen, $\mathcal{E}, \mathcal{D}$) *is IND-CPA secure.*

Proof. We will prove that $\mathcal{A}$ can not do better than guess the value of b in the delayed ballot privacy experiment via a series of hybrid games. We exploit the delay on the bulletin board and also take advantage of the fact that in a referendum the number of possible results is linear in the number of votes.

We will focus on the **VoteOracle** calls that the adversary makes, as that is where the experiment diverges depending on b. Let q_v,q_b be upper bounds on the number of **VoteOracle** and **BallotOracle** queries made by $\mathcal{A}$ for a particular security parameter λ. Let $q_\Sigma = q_v + q_b$ be the total number of queries.

We define as $\mathbf{Exp}^2$ the experiment $\mathbf{Exp}^{BP}_{\Pi^{REF},\mathcal{A}}$ where all **VoteOracle** calls produce a ballot with a simulated proof π instead of a real one. We also define a series of hybrid games H^1_i for $i \in \{0, q_v\}$ in which the first i **VoteOracle** calls produce a ballot with a simulated proof π instead of a real one. Via a straightforward hybrid argument, if $\mathcal{A}$ can distinguish between $\mathbf{Exp}^2 = H^1_{q_v}$ and $\mathbf{Exp}^{BP}_{\Pi^{REF},\mathcal{A}} = H^1_0$ with a non-negligible probability there must be a value of i such that he can distinguish H^1_i and H^1_{i+1}. This contradicts the honest verifier zero knowledge property of $(\mathcal{G}, \mathcal{P}, \mathcal{V})$, since for all i, H^1_i and H^1_{i+1} differ (at most) only in a single proof transcript. Thus $\mathcal{A}$ wins $\mathbf{Exp}^2$ with probability negligibly close to $\mathbf{Exp}^{BP}_{\Pi^{REF},\mathcal{A}}$.

We also define a series of hybrid games H^2_i for $i \in \{0, q_v\}$ as $\mathbf{Exp}^2$ in which the first i **VoteOracle** calls operate as if $b = 1$ and the rest as if $b = 0$. If $\mathcal{A}$ can win $\mathbf{Exp}^2$ with non-negligible probability, he can distinguish between H^2_0 and $H^2_{q_v}$ and thus there must be a value of i such that $\mathcal{A}$ can distinguish H^2_i and H^2_{i+1}.

Let the variable *RES* be the sum of the votes contained in ballots with correct proofs which appear in the bulletin board BB' before **CheckBoard** is called. We note that *RES* only takes values in $\{0, \ldots, q_\Sigma\}$.

Let $p(\lambda)$ be a polynomial such that $\mathcal{A}$ can distinguish between H^2_i and H^2_{i+1} with probability at least $\frac{1}{2} + \frac{1}{p(\lambda)}$ for an infinite number of $\lambda \in \mathbb{N}$. We will construct an adversary

$\mathcal{B}$ that can obtain non-negligible advantage against the IND-CPA security of $(\mathcal{K},\mathcal{E},\mathcal{D})$ by simulating an election against $\mathcal{A}$. Initially $\mathcal{B}$ obtains a public key ek from the IND-CPA experiment and completes the setup as normal, obtaining (pk,vk) from $\mathcal{G}$. $\mathcal{B}$ proceeds by following Π^{REF} with the following difference: the first i **VoteOracle** calls operate as if $b = 1$. The next **VoteOracle** (which we can assume w.l.o.g to be $v^* = 1$) is answered as if it was successful, but $\mathcal{B}$ does not update BB. Afterwards, before **CheckBoard** is called, the experiment is suspended and the state st of $\mathcal{A}$ is saved along with the bulletin boards and keys as $\sigma = (ek,(pk,vk),st,BB,BB')$. $\mathcal{B}$ does not know the value of RES, but knows that it takes values in $\{0,\dots,q_\Sigma\}$.

Let $\mathbf{Exp}^3(\sigma,v,r)$, where $\sigma = (ek,(pk,vk),st,BB,BB')$ be the following experiment: Produce B^* as a fresh ballot (with simulated proof) containing v, add B^* to BB, restore the adversary's state to st and resume the voting protocol starting at **CheckBoard**. The **Tally** query is answered with r. The result of the experiment is 1 only if $v = \hat{b}$ where $\hat{b}$ is $\mathcal{A}$'s reply.

We note when $r = RES$, $\mathbf{Exp}^3(\sigma,v,r)$ produces the same output as H_i^2 or H_{i+1}^2 depending only on v. We call a saved state σ "good" if $\mathcal{A}$ has a non-negligible advantage in distinguishing $\mathbf{Exp}^3(\sigma,0,RES)$ from $\mathbf{Exp}^3(\sigma,1,RES)$, where RES is determined uniquely by BB'. Because $\mathcal{A}$ can distinguish between H_i^2 and H_{i+1}^2, a state created by $\mathcal{B}$ must be "good" with non-negligible probability.

$\mathcal{B}$ repeats the following $\lambda \cdot p(\lambda)$ times: run $\mathbf{Exp}^3(\sigma,v,r)$ for all combinations of $v \in \{0,1\}$ and $r \in \{0,\dots,q_\Sigma\}$.

Afterwards, $\mathcal{B}$ can determine the value of r for which $\mathcal{A}$ has best distinguished between $v = 0$ and 1. We note that because the true value of RES is included in the iterations, if σ is a "good" state, there is a value of r for which $\mathcal{A}$ achieves at least $\frac{1}{p(\lambda)}$ advantage in distinguishing v, and after $\lambda \cdot p(\lambda)$ experiments Chernoff-bounds show that $\mathcal{B}$ has a good estimate of $\mathcal{A}$'s advantage for each value of r.

After determining the optimal value of r, $\mathcal{B}$ will send $(0,1)$ to the IND-CPA experiment and obtain a challenge ciphertext $\hat{c}$. $\mathcal{B}$ produces a simulated proof $\hat{\pi}$ for c, adds $\hat{B}$ to the saved board and resumes the experiment a final time. $\mathcal{B}$ finally forwards the reply of A to the IND-CPA experiment.

Because $\mathcal{B}$ will proceed without restarting with non-negligible probability, it runs in expected polynomial time. Because with overwhelming probability $\mathcal{B}$ proceeds only when

Reference	Elements			Size (bits)			Total (bits)
	$\mathbb{G}$	$\mathbb{F}$	$\mathbb{G}_L$	$\mathbb{G}$.	$\mathbb{F}$.	$\mathbb{G}_L$.	
ElGamal (ECC)	4	3		256	256		1.75K
ElGamal ($\mathbb{Z}_p$)	4	3		4.4K	256		18.5K
This work (naive)	3		3	4.4K		15K	58.2K
This work (baseline)	3		3	4.4K		11.6K	48K

Table 4.1: Size comparison between our work and baseline Helios - ElGamal, for parameters targeting 128 bit security.

σ is a “good” state, the advantage in distinguishing whether c contains 0 or 1 is non-negligible. □

4.4.3 Efficiency

In this section, we compare the size of a simple 0/1 ballot for our scheme, compared to a standard Helios instantiation with ElGamal. We use the guidelines from [Len05] and target 128 bit security. In both cases, we give the cost in terms of group elements $\mathbb{G}$, field elements $\mathbb{F}$ and large group elements $\mathbb{G}_L$. Large group elements are used for the outer encryption layer of our DFN-based construction (i.e the keys in).

The three variants of our instantiation differ as follows: In the the naive version, p and q increase in size proportionally. In the baseline version, only p increases in size, as the size of q is already large enough and the size of the message space depends only on p.

Chapter 5

Efficient Protocols for Arithmetic Circuits

In the previous chapter, we gave a protocol for the satisfiability of a binary circuit. While simple conceptually, it was not particularly efficient as we relied on a separate subprotocol for every multiplication gate and wire. In this chapter we will construct a much more efficient but also more complex protocol that is tailored to arithmetic circuits. The additional complexity means that the DFN transformation is no longer applicable: some aspects (e.g. multiple rounds or answers linear in higher powers of the challenge) could be handled by generalising the transformation, but others (like answers involving group elements with the challenge used as an exponent) would require a large jump in its expressive power. This leaves us with either a multi-round argument or using the Fiat-Shamir heuristic in the random oracle model.

With regards to efficiency, we provide a zero-knowledge argument for arithmetic circuit satisfiability with a communication complexity that grows logarithmically in the size of the circuit. The round complexity is also logarithmic and for an arithmetic circuit with fan-in 2 gates the computation of the prover and verifier is linear in the size of the circuit. The soundness of our argument relies solely on the well-established discrete logarithm assumption in prime order groups.

At the heart of our new argument system is an efficient zero-knowledge argument of knowledge of openings of two Pedersen multicommitments satisfying an inner product relation, which is of independent interest. The inner product argument requires logarithmic communication and interaction, and linear computation for both the prover and the verifier.

We also develop a scheme to commit to a polynomial and later reveal the evaluation at an arbitrary point, in a verifiable manner. This is used to build an optimised version of the constant round square root complexity argument of Groth [Gro09b], reducing both communication and round complexity.

The research presented in this chapter was joint work with Jonathan Bootle, Andrea Cerulli, Jens Groth and Christophe Petit, and published in [BCC^{+}16b]. The author contributed to the investigation in the recursive version of the main protocol and in the idea of utilising the homomorphic property of the bases in our commitment scheme. Additionally, in the root version of the protocol, and specifically in using the constraints system required by the multiplication argument to handle additions "for free". The author also contributed the Python implementation of the protocol, together with Andrea Cerulli as well as the benchmarking.

5.1 Introduction

Our goal is to build an efficient argument system for the satisfiability of an arithmetic circuit, i.e., a circuit that consists of addition and multiplication gates over a finite field $\mathbb{Z}_p$. Moreover we want to base the security of this argument solely on the discrete logarithm assumption: this will provide both strong security guarantees and good efficiency since there exists no known attacks better than generic ones for well-chosen elliptic curve subgroups.

The most efficient zero-knowledge arguments solely based on the discrete logarithm assumption are Groth's protocol based on linear algebra [Gro09b] and its variant by Seo [Seo11]. Both of these protocols have a communication complexity that is proportional to the square root of the circuit size. This square root complexity has since then appeared (Sect. 2.1.2) as a (perhaps fundamental) barrier for discrete logarithm-based arguments for circuit satisfiability in this setting.

5.1.1 Contributions

We provide an honest verifier zero-knowledge argument for arithmetic circuit satisfiability based on the discrete logarithm assumption that only requires a *logarithmic* communication complexity. Our argument has perfect completeness and perfect special honest verifier zero-knowledge. Soundness is computational and based on the discrete logarithm assumption. We require a logarithmic number of moves, and both the prover and verifier have linear computational complexity. The argument is therefore efficient on all parameters with the biggest improvement being in the communication complexity.

Improved Square Root Complexity Argument. We start from the circuit satisfiability argument of Groth [Gro09b], which requires 7 moves and has square root communication complexity in the *total* number of gates. In this argument the prover commits to all the wires using homomorphic multicommitments, verifies addition gates using the homomorphic properties, and uses a product argument to show that the multiplication gates are satisfied.

We first improve Groth's argument into a 5 moves argument with square root communication complexity in the number of *multiplication gates* only. We achieve fewer moves compared to [Gro09b] by avoiding generic reductions to linear algebra statements. We remove the communication cost of the addition gates in the argument by providing a technique that can directly handle a set of Hadamard products and linear relations together. Another efficiency improvement is a subroutine to commit to a polynomial and later reveal its evaluation at an arbitrary point in a verifiable manner. In Section 5.4 we provide a protocol to perform this task, which has a square root communication complexity with respect to the degree of the polynomial, and which may be of independent interest.

Logarithmic Complexity Argument. In spite of all these improvements, the above argument still requires a square root communication complexity with respect to multiplication gates. In the first move the prover commits to all circuit wires using $3m$ commitments to n elements each, where $mn = N$ is a bound on the number of multiplication gates, and in the last move after receiving a challenge he opens one commitment that can be constructed from the previous ones and the challenge. By setting $m \approx n$ we get a minimal communication complexity of $O(\sqrt{N})$.

Our key idea to break this square root communication complexity barrier is to replace the last opening step in this protocol by an argument of knowledge of the opening values. Using specific properties of Pedersen multicommitments, namely homomorphic properties with respect to the keys, we rewrite this argument as an argument of knowledge of openings of two homomorphic commitments, satisfying an inner product relation. In Section 5.5 we provide an argument system for this problem, which only requires a logarithmic communication with respect to the vector sizes. The argument is built in a recursive way, reducing the size and complexity of the statement further in each recursion step. Using this inner product argument as a subroutine we obtain an arithmetic circuit satisfiability argument with logarithmic communication.

Implementation. In Section 5.7 we report on an implementation of our arguments. To show the practicality of our results we compare the efficiency of our implementation to that of Pinocchio [PHGR13]. Pinocchio is a practical verifiable computation scheme allowing a constrained client to outsource computation of a function to a powerful worker and to efficiently verify the outcome of the function. It uses quadratic arithmetic programs, a generalisation of arithmetic circuits, and for some functions achieves verification that is faster than local computation. While we do not achieve comparably fast verification, we compare favourably in terms of prover computation, and do so under simpler assumptions.

5.2 Related Work

Table 5.1 compares the most efficient previous zero-knowledge arguments based on the discrete logarithm assumption with our scheme, when allowing for 5 moves or a logarithmic number of moves. Using 5 moves, our scheme requires significantly less computation

Reference	Moves	Communication		Prover Complexity		Verifier Complexity	
		$\mathbb{G}$	$\mathbb{Z}_p$	exp.	mult.	exp.	mult.
[CD98]	3	$6N$	$5N+2$	$6N$	$6N$	$6N$	0
[Gro09b]	7	$9\sqrt{N}+4$	$7\sqrt{N}+6$	$\frac{6N}{\log N}$	$O(N\log N)$	$\frac{39\sqrt{N}}{\log N}$	$O(N)$
[Gro09b]	$2\log N+5$	$2\sqrt{N}$	$7\sqrt{N}$	$\frac{6N}{\log N}$	$O(N)$	$\frac{18\sqrt{N}}{\log N}$	$O(N)$
[Seo11]	5	$30\sqrt{N}$	$7\sqrt{N}$	$\frac{6N}{\log N}$	$O(N\log N)$	$\frac{77\sqrt{N}}{\log N}$	$O(N)$
This work	5	$2\sqrt{N}$	$2\sqrt{N}$	$\frac{6N}{\log N}$	$3N\log N$	$\frac{8\sqrt{3N}}{\log N}$	$O(N)$
This work	$2\log N+1$	$4\log N+7$	$2\log N+6$	$12N$	$O(N)$	$4N$	$O(N)$

Table 5.1: Efficiency comparison between our arguments and the most efficient interactive zero-knowledge arguments relying on discrete logarithm. We express communication in number of group elements $\mathbb{G}$ and field elements $\mathbb{Z}_p$ and computation costs in number of exponentiations over $\mathbb{G}$ and multiplications over $\mathbb{Z}_p$. The efficiency displayed is for a circuit with N multiplication gates.

than [Seo11]. On the other hand when using a logarithmic number of moves and applying a reduction similar to [BG12], our scheme dramatically improves the communication costs with respect to all previous work without incurring any significant overhead. We note that [BG12] uses the reduction to reduce computation whereas we use it to reduce communication.

As part of our construction we give a protocol for committing to a polynomial and later revealing an evaluation of the polynomial in a given point. Kate et al. [KZG10] have also provided protocols to commit to polynomials and then evaluate them at a given point in a verifiable way. Their protocols only require a constant number of commitments but security relies on pairing assumptions. Our polynomial commitment protocol has square root communication complexity but relies solely on the discrete logarithm assumption.

5.3 Preliminaries

Throughout the chapter we let $\mathbb{G}$ be a group of prime order p. Let $\vec{g} = (g_1,\ldots,g_n) \in \mathbb{G}^n$ and $\vec{f} = (f_1,\ldots,f_n) \in \mathbb{Z}_p^n$. We write $\vec{g}^{\vec{f}}$ for the multi-exponentiation $\vec{g}^{\vec{f}} = \prod_{i=1}^n g_i^{f_i}$. A multi-exponentiation of size n can be computed at a cost of roughly $\frac{n}{\log n}$ single group exponentiations using the multi-exponentiation techniques of [Lim00, Möl01, MR08].

5.3.1 Arithmetic Circuits

Our satisfiability arguments consider arithmetic circuits described as a list of multiplication gates together with a set of linear consistency equations relating the inputs and outputs of the gates. In this section, we show how to reduce an arbitrary arithmetic circuit to this format.

An arithmetic circuit over a field $\mathbb{Z}_p$ and variables $(a_1, \ldots, a_m)$ is a directed acyclic graph whose vertices are called gates. Gates of in-degree 0 are inputs to the circuit and labelled with some a_i or a constant field element. All other gates are labelled $+$ or $\times$. We may consider fan-in 2 circuits, in which case all of the $+$ and $\times$ gates have in-degree 2, or arbitrary fan-in circuits.

Arithmetic circuits can be described alternatively as a list of multiplication gates with a collection of linear consistency equations relating the inputs and outputs of the gates. Our zero-knowledge protocols for circuit satisfiability use circuits in this form. Any circuit described as an acyclic graph can be efficiently converted into the alternative description.

We show how to remove addition and multiplication-by-constant gates from an arithmetic circuit A, and replace them with bilinear consistency equations on the inputs and outputs of the remaining gates, such that satisfiability of the equations is equivalent to satisfiability in the original circuit.

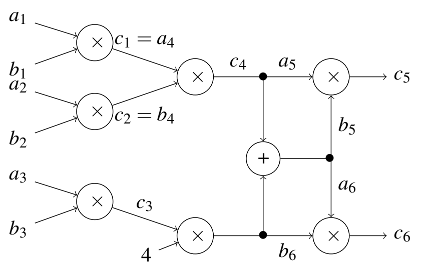
Let B be the sub-circuit of A containing all wires and gates before a multiplication gate, with m input wires and n output wires. Label the m inputs of B with the unit vectors $\vec{e}_i = (0, \ldots, 1, \ldots, 0)$ of length m. For every addition gate with inputs labelled as $\vec{x}, \vec{y}$, label the output wire as $\vec{x} + \vec{y}$. For every multiplication-by-constant gate with inputs $\vec{x}$ and constant c label the output with $c\vec{x}$. By proceeding inductively, the n outputs of B are now labelled with vectors of length m representing them as linear combinations of the inputs.

This requires at most $m|B|$ arithmetic operations. Note however that all outputs of B are linear combinations of the inputs, and that B can be written with $n(2m-1)$ fan-in 2 gates in such a way that the consistency equations can be trivially read off from the circuit description. More specifically, a linear combination $\sum_{i=1}^{m} a_i x_i$ can be produced using m multiplication-by-constant gates and $m-1$ addition gates to add the answers together.

We can now remove the gates of B from A. We also remove any multiplication gates whose inputs are the inputs of the new circuit. Now we simply repeat the process of finding consistency equations until we have considered the whole of A. In Figure 5.1 there is an example of a circuit together and the corresponding consistency equations.

The first (input) and final (output) sub-circuits require additional processing. We show how to do this for the output sub-circuit. The input sub-circuit is very similarly handled.

Let B be the output sub-circuit. Write $(a_1, \ldots, a_m) = \vec{a}$ for the input wires of B and $(b_1, \ldots, b_n) = \vec{b}$ for the output wires. Without loss of generality, we may ignore variable



$$\begin{aligned} c_1 &= a_4 \\ c_2 &= b_4 \\ c_4 &= a_5 \\ 4c_3 + c_4 &= b_5 \\ 4c_3 + c_4 &= a_6 \\ 4c_3 &= b_6 \end{aligned}$$

Figure 5.1: A simple arithmetic circuit, and the corresponding consistency equations. The first sub-circuit contains the wires $a_1, b_1, c_1, a_2, b_2, c_2, a_3, a_3, c_3$. The second sub-circuit contains the wires $c_1, a_4, c_2, b_4, c_4, c_5, c_6$. The third sub-circuit B contains the wires c_3, c_4, a_5, b_5, a_6.

output wires. By construction of B, each output b_i is of the form $\sum_{i=1}^{n} q_{ij} a_j + p_i$, with consistency equations obtained as above. We write this in terms of an $m \times n$ matrix Q and a column vector $\vec{p}$ of size m, namely

$$\vec{b} = \mathsf{Q}\vec{a} + \vec{p}.$$

Let r be the rank of Q. We convert Q into reduced row echelon form R, writing

$$\vec{b''} = \mathsf{R}\vec{a}.$$

By the properties of reduced row echelon form, after relabelling the a_i and permuting the columns of R to match, we have that $b''_i = a_i + \sum_{j=l+1}^{m} r_{ij} a_j$ for $1 \leq i \leq l$. Therefore, we may consider $a_{l+1}, \ldots, a_m$ as free wires and express other a_i as linear functions of these wires plus constants.

Note that if $b''_i \neq 0$ for some $i > l$, the circuit can never be satisfied anyway. However, assuming that our statement is a satisfiable circuit, with a witness consisting of satisfying wire values, this never occurs. Then the original circuit is satisfied if and only if the a_i values satisfy the consistency equations.

If Q is an $m \times n$ matrix then it can be converted into reduced row echelon form using $O(\max(m,n)mn)$ operations. It is trivial that $m \leq 2|B|$ and $n \leq |B|$. This gives an upper bound of $O(|B|^3)$ computation for the output sub-circuit. Note that this is often a large over-estimate; this upper bound occurs for circuits of depth 1 where inputs feed into distinct gates. For circuits of large depth, where the same input is fed into several gates, the upper bound will definitely not be reached.

The case of the input sub-circuit is very similar, except that we take the transpose of the matrix.

5.3.2 Full zero knowledge & non-interactivity

In real life applications special honest verifier zero knowledge may not suffice since a malicious verifier may give non-random challenges. However, it is easy to convert an SHVZK argument into a full zero-knowledge argument secure against *arbitrary* verifiers in the common reference string model using standard techniques [Gro04b, GMY06] . The conversion can be very efficient and only costs a small additive overhead.

The Fiat-Shamir transformation (Sect. 3.4.6) can be applied to our arguments to make them non-interactive, as well as full zero knowledge at the cost of using the random oracle model in the security proofs. From an efficiency point of view this is especially useful for the arguments in Sections 5.5 and 5.6.4, reducing a logarithmic number of moves to a single one.

5.4 Commitments to Polynomials

In this section, we present a protocol to commit to a polynomial $t(X)$ and later reveal the evaluation of $t(X)$ at any point $x \in \mathbb{Z}_p^*$ together with a proof that enables a verifier to check that the evaluation is correct with respect to the committed $t(X)$. We will consider Laurent polynomials $t(X) \in \mathbb{Z}_p[X, X^{-1}]$ i.e. polynomials in which we allow terms of negative degree. This protocol will be used as a subroutine for the arguments described in Sections 5.6.3 and 5.6.4.

A simple solution for this problem would be to send commitments to coefficients of $t(X)$ individually, from which the evaluation of $t(X)$ at any particular point can be verified using the homomorphic properties. This solution requires d group elements to be sent, where d is the number of non-zero coefficients in $t(X)$. As we shall show it is possible to reduce the communication costs to $O(\sqrt{d})$ group elements, where $d = d_2 + d_1$ if $t(X) = \sum_{k=-d_1}^{d_2} t_k X^k$.

For clarity we first informally describe our protocol for a standard (not Laurent) polynomial $t(X) = \sum_{k=0}^{d} t_k X^k$. We then extend this informal description to Laurent polynomials with zero constant term. We finally provide a formal description of the protocol and analyze its security and efficiency.

5.4.1 Main idea for standard polynomials.

Let $t(X) = \sum_{k=0}^{d} t_k X^k$ be a polynomial with coefficients in $\mathbb{Z}_p$ and assume $d+1 = mn$. We can write $t(X) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} t_{i,j}(X) X^{in+j}$ and arrange the coefficients in a $m \times n$ matrix

$$\begin{pmatrix} t_{0,0} & t_{0,1} & \cdots & t_{0,n-1} \\ t_{1,0} & t_{1,1} & \cdots & t_{1,n-1} \\ \vdots & \vdots & & \vdots \\ t_{m-1,0} & t_{n-1,1} & \cdots & t_{m-1,n-1} \end{pmatrix}$$

Now, $t(X)$ can be evaluated by multiplying the matrix by row and column vectors.

$$t(X) = \begin{pmatrix} 1 & X^n & \cdots & X^{(m-1)n} \end{pmatrix} \begin{pmatrix} t_{0,0} & t_{0,1} & \cdots & t_{0,n-1} \\ t_{1,0} & t_{1,1} & \cdots & t_{1,n-1} \\ \vdots & \vdots & & \vdots \\ t_{m-1,0} & t_{n-1,1} & \cdots & t_{m-1,n-1} \end{pmatrix} \begin{pmatrix} 1 \\ X \\ \vdots \\ X^{n-1} \end{pmatrix}$$

The idea behind the protocol is to commit to the rows of this matrix using commitments $T_0, \ldots, T_{m-1}$. Later, when given an evaluation point $x \in \mathbb{Z}_p$ we can use the homomorphic property of the commitment scheme to compute the commitment $\prod_{i=0}^{m-1} T_i^{x^{in}}$ to the vector

$$\bar{\vec{t}} = \begin{pmatrix} 1 & x^n & \cdots & x^{(m-1)n} \end{pmatrix} \begin{pmatrix} t_{0,0} & t_{0,1} & \cdots & t_{0,n-1} \\ t_{1,0} & t_{1,1} & \cdots & t_{1,n-1} \\ \vdots & \vdots & & \vdots \\ t_{m-1,0} & t_{m-1,1} & \cdots & t_{m-1,n-1} \end{pmatrix}$$

The prover opens this latter commitment and now it is easy to compute $v = t(x)$ from $\bar{\vec{t}}$ and x.

The problem with this straightforward solution is that it leaks partial information about the coefficients of $t(X)$. We remedy this by inserting some blinding values $u_1, \ldots, u_{n-1}$ to hide the weighted sum of the coefficients in each column. However, we make sure that the blinding values cancel each other out so that we still get the correct evaluation of the polynomial. More precisely, we commit to the rows of the following $(m+1)\times n$ matrix

$$T = \begin{pmatrix} t_{0,0} & t_{0,1}-u_1 & \cdots & t_{0,n-2}-u_{n-2} & t_{0,n-1}-u_{n-1} \\ t_{1,0} & t_{1,1} & \cdots & t_{1,n-2} & t_{1,n-1} \\ \vdots & \vdots & & & \vdots \\ t_{m-1,0} & t_{m-1,1} & \cdots & t_{m-1,n-2} & t_{m-1,n-1} \\ u_1 & u_2 & \cdots & u_{n-1} & 0 \end{pmatrix}$$

with U being a commitment to the last row. This time

$$t(X) = \begin{pmatrix} 1 & X^n & \cdots & X^{(m-1)n} & X \end{pmatrix} T \begin{pmatrix} 1 \\ X \\ X^2 \\ \vdots \\ X^{n-1} \end{pmatrix}$$

We now open $U^x \prod_{i=0}^{m-1} T_i^{x^{in}}$ by revealing the vector

$$\bar{\vec{t}} = \begin{pmatrix} 1 & x^n & \cdots & x^{(m-1)n} & x \end{pmatrix} T$$

This still allows us to compute $t(x)$, but due to the blinders we no longer leak information about the coefficients of $t(X)$. In fact, each element of $\bar{\vec{t}}$ is uniformly random, conditional on their weighted sum being equal to $t(x)$, which the prover intends for the verifier to learn anyway.

5.4.2 Extension to Laurent polynomials.

Let now $t(X)$ be a Laurent polynomial $t(X) = \sum_{i=-d_1}^{d_2} t_i X^i$ with constant term $t_0 = 0$. Let m_1, m_2, n be positive integers such that $d_1 = nm_1$ and $d_2 = nm_2$ and write $t(X) = X^{-m_1 n} t'(X) + X t''(X)$ for degree $d_1 - 1$ and $d_2 - 1$ polynomials $t'(X), t''(X) \in \mathbb{Z}_p[X]$. We can write $t'(X) = \sum_{i=0}^{m_1-1}\sum_{j=0}^{n-1} t'_{i,j} X^{in+j}$ and $t''(X) = \sum_{i=0}^{m_2-1}\sum_{j=0}^{n-1} t''_{i,j} X^{in+j}$.

We can arrange the coefficients of $t'(X)$ and $t''(X)$ in a $(m_1+m_2)\times n$ matrix T. We commit to both $t'(X)$ and $t''(X)$ simultaneously by committing to the rows of the matrix using commitments T'_i and T''_i. As when committing to polynomials we add blinders

$u_1, \ldots, u_{n-1}$ and make a commitment U to the additional last row arising from this.

$$T = \begin{pmatrix} t'_{0,0} & t'_{0,1} & \cdots & t'_{0,n-1} \\ t'_{1,0} & t'_{1,1} & \cdots & t'_{1,n-1} \\ \vdots & \vdots & & \vdots \\ t'_{m_1-1,0} & t'_{m_1-1,1} & \cdots & t'_{m_1-1,n-1} \\ t''_{0,0} & t''_{0,1} - u_1 & \cdots & t''_{0,n-1} - u_{n-1} \\ t''_{1,0} & t''_{1,1} & \cdots & t''_{1,n-1} \\ \vdots & \vdots & & \vdots \\ t''_{m_2-1,0} & t''_{m_2-1,1} & \cdots & t''_{m_2-1,n-1} \\ u_1 & u_2 & \cdots & 0 \end{pmatrix} = \begin{pmatrix} \vec{t}'_0 \\ \vec{t}'_1 \\ \vdots \\ \vec{t}'_{m_1-1} \\ \vec{t}''_0 \\ \vec{t}''_1 \\ \vdots \\ \vec{t}''_{m_2-1} \\ \vec{u} \end{pmatrix}$$

Define vectors

$$\vec{Z} = \vec{Z}(X) = \left(X^{-m_1 n}, X^{-(m_1-1)n}, \ldots, X^{-n}, X, X^{n+1}, \ldots, X^{(m_2-1)n+1}, X^2\right)$$

$$\vec{X} = \vec{X}(X) = \begin{pmatrix} 1 \\ X \\ \vdots \\ X^{n-1} \end{pmatrix}$$

and we have $t(X) = \vec{Z}T\vec{X}$.

To evaluate at $x \in \mathbb{Z}_p^*$ we open $\left(\prod_{i=0}^{m_1-1} (T'_i)^{x^{(i-m_1)n}}\right) \left(\prod_{i=0}^{m_2-1} (T''_i)^{x^{in+1}}\right) U^{x^2}$ to the vector $\bar{\vec{t}} = \vec{Z}(x)T$. This allows us to compute $t(x)$ as $\bar{\vec{t}}\vec{X}(x)$. The blinders hide the weighted sums of each column as before, and now the verifier is able to compute $t(x)$ without gaining additional information about its coefficients.

5.4.3 Evaluation Protocol.

Our protocol is made of the following three algorithms.

- PolyCommit$(ck, m_1, m_2, n, t(X)) \rightarrow (\mathsf{pc}, \mathsf{st})$**:** Take as input a commitment key ck and a Laurent polynomial $t(X) = \sum_{i=-m_1 n}^{nm_2} t_i X^i$ with constant coefficient $t_0 = 0$. Pick blinders $u_1, \ldots, u_{n-1} \leftarrow \mathbb{Z}_p$ and randomness $\tau_u, \tau'_0, \ldots, \tau'_{m_1-1}, \tau''_0, \ldots, \tau''_{m_2-1} \leftarrow \mathbb{Z}_p$. Set $\vec{\tau} = \left(\tau'_0, \ldots, \tau'_{m_1-1}, \tau''_0, \ldots, \tau''_{m_2-1}, \tau_u\right)$. Compute

$$T'_i = \mathrm{Com}_{ck}(\vec{t}'_i; \tau'_i), \qquad T''_i = \mathrm{Com}_{ck}(\vec{t}''_i; \tau''_i), \qquad U = \mathrm{Com}_{ck}(\vec{u}; \tau_u)$$

Return a polynomial commitment $\mathsf{pc} = \left(\{T_i'\}_{i=0}^{m_1-1}, \{T_i''\}_{i=0}^{m_2-1}, U\right)$ and private information $\mathsf{st} = (t(X), \vec{\tau})$.

- PolyEval$(\mathsf{st}, x) \to \mathsf{pe}$**:** Compute

$$\bar{\vec{t}} = \vec{Z}(x)T, \qquad \bar{\tau} = \vec{Z}(x) \cdot \vec{\tau}$$

Return $\mathsf{pe} = (\bar{\vec{t}}, \bar{\tau})$.

- PolyVerify$(ck, m_1, m_2, n, \mathsf{pc}, \mathsf{pe}, x) \to v$**:** The verifier checks whether

$$\mathrm{Com}_{ck}(\bar{\vec{t}}; \bar{\tau}) = \left(\prod_{i=0}^{m_1-1} (T_i')^{x^{(i-m_1)n}}\right) \left(\prod_{i=0}^{m_2-1} (T_i'')^{x^{in+1}}\right) U^{x^2}$$

If the check is satisfied the verifier returns $v = t(x) = \bar{\vec{t}}\vec{X}(x)$.

Otherwise, the verifier rejects pe as invalid with respect to pc and x and returns $v = \perp$.

5.4.4 Security Properties.

We define three security properties for our protocol: completeness, ℓ-special soundness, and special-honest-verifier zero-knowledge. Later, the protocol is used as a sub-protocol inside our zero-knowledge arguments-of-knowledge. We opt out of using the definitions of [KZG10] as our context is somewhat narrow (e.g. we do not intent to reveal an entire polynomial) so we opt to only model the security properties we require. These properties will help us to prove the completeness, witness-extended emulation, and special honest verifier zero knowledge for the zero knowledge argument.

The definition of completeness simply guarantees that if PolyCommit, PolyVerify are carried out honestly, then PolyVerify will accept and return a commitment to the evaluation of the polynomial.

Definition 22 (Perfect Completeness)**.** (PolyCommit, PolyEval, PolyVerify) *has perfect completeness if for all non-uniform polynomial time adversaries $\mathcal{A}$*

$$\Pr\left[\begin{array}{l} (ck, m_1, m_2, n, t(X), x) \leftarrow \mathcal{A}(1^\lambda) \\ (\mathsf{pc}, \mathsf{st}) \leftarrow \mathrm{PolyCommit}(ck, m_1, m_2, n, t(X)) \\ \mathsf{pe} \leftarrow \mathrm{PolyEval}(\mathsf{st}, x) \\ v \leftarrow \mathrm{PolyVerify}(ck, m_1, m_2, n, \mathsf{pc}, \mathsf{pe}, x) \end{array} : v = t(x)\right] = 1$$

where ck is a key for a homomorphic commitment scheme, $t(X)$ is a Laurent polynomial of degrees $d_1 = m_1 n, d_2 = m_2 n$ and $x \in \mathbb{Z}_p^*$.

The definition of ℓ-Special Soundness says that given ℓ accepting evaluations for different evaluation points, but from the same commitment pc, then it is possible to extract either a valid Laurent polynomial $t(X)$ with zero constant term that is consistent with the evaluations produced or a breach in the binding property of the commitment scheme. Furthermore, any other accepting evaluations for the same commitment will also be evaluations of $t(X)$.

Definition 23 (Statistical ℓ-Special Soundness)**.** $(\text{PolyCommit}, \text{PolyEval}, \text{PolyVerify})$ *is statistically ℓ-special sound if there exists a probabilistic polynomial time algorithm χ that, given ℓ accepting transcripts with the same commitment* pc*, either extracts the committed polynomial $t(X)$, or extracts a break of the binding property of the underlying commitment scheme. For all adversaries $\mathcal{A}$ and all $L \geq \ell$*

$$\Pr\left[\begin{array}{lcc} ck \leftarrow \text{CGen}(1^\lambda) & & \forall i:\ v_i = \vec{Z}(x_i)T\vec{X}(x_i) \\ (m_1, m_2, n, \mathsf{pc}, x_1, \mathsf{pe}_1, \ldots, x_L, \mathsf{pe}_L) \leftarrow \mathcal{A}(ck) & & \text{or } \exists j \text{ s.t.} \\ \text{Parse } \mathsf{pe}_i = (\vec{\bar{t}}_i, \bar{\tau}_i) & : & \text{Com}_{ck}(\vec{\bar{t}}_j; \bar{\tau}_j) = \\ (T, \vec{\tau}) \leftarrow \chi(ck, m_1, m_2, n, \mathsf{pc}, x_1, \mathsf{pe}_1, \ldots, x_l, \mathsf{pe}_l) & & \text{Com}_{ck}\left(\vec{Z}(x_j)T; \vec{Z}(x_j)\vec{\tau}\right), \\ v_i \leftarrow \text{PolyVerify}(ck, m_1, m_2, n, \mathsf{pc}, \mathsf{pe}_i, x_i) & & \text{where } \vec{\bar{t}}_j \neq \vec{Z}(x_j)T \end{array}\right] \approx 1,$$

where $x_1, \ldots, x_\ell$ are distinct, $x_i \in Z_p^*$, $\mathsf{pe}_i \in \mathbb{Z}_p^n \times \mathbb{Z}_p$, $T \in \mathbb{Z}_p^{(m_1+m_2)\times n}$, and $\vec{\tau} \in \mathbb{Z}_p^{m_1+m_2}$.

Perfect special honest verifier zero-knowledge means that given any value v and evaluation point x, it is possible to simulate pc and pe, distributed exactly as in a real execution of the protocol where v was the evaluation of $t(X)$ at x.

Definition 24 (Perfect Special Honest Verifier Zero Knowledge)**.** (PolyCommit, PolyEval, PolyVerify) *has* perfect special honest verifier zero knowledge (SHVZK) *if there exists a probabilistic polynomial time simulator $\mathcal{S}$ such that for all interactive non-uniform* *polynomial time adversaries $\mathcal{A}$*

$$\Pr\left[\begin{array}{l}(ck,m_1,m_2,n,t(X),x)\leftarrow\mathcal{A}(1^\lambda)\\ (\mathsf{pc},\mathsf{st})\leftarrow\mathrm{PolyCommit}(ck,m_1,m_2,n,t(X))\\ \mathsf{pe}\leftarrow\mathrm{PolyEval}(\mathsf{st},x)\end{array} : \mathcal{A}(\mathsf{pc},\mathsf{pe})=1\right]$$
$$=\Pr\left[\begin{array}{l}(ck,m_1,m_2,n,t(X),x)\leftarrow\mathcal{A}(1^\lambda)\\ (\mathsf{pc},\mathsf{pe})\leftarrow\mathcal{S}(ck,m_1,m_2,n,x,t(x))\end{array} : \mathcal{A}(\mathsf{pc},\mathsf{pe})=1\right]$$

where ck is a key for a homomorphic commitment scheme, $t(X)$ is a Laurent polynomial of degrees $d_1=m_1n, d_2=m_2n$ and $x\in\mathbb{Z}_p^*$.

Theorem 6. *The polynomial commitment protocol has perfect completeness, perfect special honest verifier zero-knowledge and $(m_1+m_2)n+1$-special soundness for extracting either a breach of the binding property of the commitment scheme or openings to the polynomial.*

Proof. Direct verification of the protocol shows that the verifier correctly obtains the evaluation of the committed polynomial $t(X)$ at the point x.

For simplicity, we prove SHVZK and special-soundness for the case of Pedersen commitments, but all aspects of the proof carry over in the general case.

For special honest verifier zero-knowledge suppose we have an evaluation point x and an evaluation v. To simulate the polynomial evaluation we first pick random $\bar{t}_2,\ldots,\bar{t}_n,\bar{\tau}\leftarrow\mathbb{Z}_p$ and $T'_0,\ldots,T'_{m_1-1},T''_0,\ldots,T''_{m_2-1}\leftarrow\mathbb{G}$. Now, $\bar{t}_1$ is computed as

$$\bar{t}_1=v-\sum_{i=2}^{n}\bar{t}_i x^{i-1}$$

Let $\bar{\vec{t}}=(\bar{t}_1,\ldots,\bar{t}_n)$ and compute

$$U=\mathrm{Com}_{ck}(\bar{\vec{t}}x^{-2};\bar{\tau}x^{-2})\left[\prod_{i=0}^{m_1-1}(T'_i)^{x^{(i-m_1)n}}\right]^{x^{-2}}\left[\prod_{i=0}^{m_2-1}(T''_i)^{x^{in+1}}\right]^{x^{-2}}$$

This is a perfect simulation. Given x,v observe that both in the simulation and in the real execution we get uniformly random $\bar{t}_2,\ldots,\bar{t}_n,\bar{\tau}\in\mathbb{Z}_p$ and $T'_0,\ldots,T'_{m_1-1},T''_0,\ldots,T''_{m_2-1}\leftarrow\mathbb{G}$. Now, conditioning on the values of x,v and the verification equations, U and $\bar{t}_1$ are uniquely determined. Therefore, the simulation is perfect, and we have special honest verifier zero-knowledge.

Finally, for special soundness note that if we have valid evaluations of $l = (m_1+m_2)n+2$ different challenges $x \in \mathbb{Z}_p^*$ the vectors $(x^{-m_1 n}, \ldots, x^{m_2 n - n+1})$ form the rows of a Vandermonde matrix multiplied by $x^{-m_1 n}$ and we can obtain any unit vector $(0, \ldots, 1, \ldots, 0)$ by taking an appropriate linear combination of these vectors. By taking the correct linear combinations of the l verification equations

$$\mathrm{Com}_{ck}(\bar{\vec{t}}; \bar{\tau}) = \left[\prod_{i=0}^{m_1-1} (T_i')^{x^{(i-m_1)n}}\right]\left[\prod_{i=0}^{m_2-1} (T_i'')^{x^{in+1}}\right] U^{x^2}$$

for these l different challenges, we can then extract an opening of any $T_i' = \mathrm{Com}_{ck}(\vec{t}_i'; \tau_i')$, $T_i'' = \mathrm{Com}_{ck}(\vec{t}_i''; \tau_i'')$ and $U = \mathrm{Com}_{ck}(\vec{u}; \tau_u)$.

This implies that whenever we have $L \geq l$ different challenges $x \in \mathbb{Z}_p^*$ with valid evaluations we must either have $\bar{\vec{t}} = \vec{Z}(x)T$ for each (x, pe) pair, where the rows of T are the extracted $\vec{t}_i'$ and $\vec{t}_i''$ and $\vec{u}$, or we could extract a violation of the binding property of the commitment scheme.

In the first case where $\bar{\vec{t}} = \vec{Z}(x)T$, for each pair, χ sets $\tau = (\tau_0', \ldots, \tau_{m_1-1}', \tau_0'', \ldots, \tau_{m_2-1}'', \tau_u)$, and sets $S, \vec{\sigma}$ to be all zeroes. Since $\bar{\vec{t}} = \vec{Z}(x)T$, it follows that $v_i = \vec{Z}(x_i)T\vec{X}(x_i)$ for each i.

Alternatively, if $\bar{\vec{t}}_i \neq \vec{Z}(x_i)T$ for some x_i and $\mathsf{pe}_i = (\bar{\vec{t}}_i, \bar{\tau}_i$, then since the verifier was accepting, we have $\mathrm{Com}_{ck}(\bar{\vec{t}}_i; \bar{\tau}_i) = \mathrm{Com}_{ck}(\vec{Z}(x_i)T; \vec{Z}(x_i)\vec{\tau})$, and this gives a violation of the binding property of the commitment scheme. □

5.4.5 Efficiency.

We now discuss the efficiency of the above protocol when instantiated with the Pedersen multicommitment scheme. The outputs pc, pe of the polynomial commitment protocol have sizes of $m_1 + m_2 + 1$ group elements and $n+1$ field elements respectively. The computational cost of computing pc is dominated by computing commitments T_i' and T_i'', corresponding to $m_1 + m_2$ n-wide multi-exponentiations. Using multi-exponentiation techniques as in [Lim00, Möl01, MR08], the total cost is roughly $\frac{(m_1+m_2)n}{\log n}$ group exponentiations. The main cost for computing pe is dominated by the $n(m_1+m_2)$ field multiplications required to compute $\vec{Z}T$. The dominant cost in PolyVerify is to check the verification equation. This costs roughly $\frac{m_1+m_2+n}{\log(m_1+m_2+n)}$ group exponentiations.

5.5 Recursive Argument for Inner Product Evaluation

We will now give an inner product argument of knowledge of two vectors $\vec{a},\vec{b} \in \mathbb{Z}_p^n$ such that $A = \vec{g}^{\vec{a}}$, $B = \vec{h}^{\vec{b}}$ and $\vec{a} \cdot \vec{b} = z$, given $z \in \mathbb{Z}_p$, A, $B \in \mathbb{G}$ and $\vec{g},\vec{h} \in \mathbb{G}^n$. The argument will be used later as a subroutine where zero-knowledge is not required, so the prover could in principle just reveal the witness $\vec{a},\vec{b}$ to the verifier. In the following we show how to use interaction to reduce the communication from linear to logarithmic in n, the length of the vectors.

The basic step in our inner product argument is a 2-move reduction to a smaller statement using techniques similar to [BG12]. It will suffice for the prover to reveal the witness for the smaller statement in order to convince the verifier about the validity of the original statement. In the full argument, prover and verifier recursively run the reduction to obtain increasingly smaller statements. The argument is then concluded with the prover revealing a witness for a very small statement. The outcome of this is a $O(\log n)$-move argument with an overall communication of $O(\log n)$ group and field elements. The inner product argument will be used in the next section to build a logarithmic size argument for circuit satisfiability.

Due to the obvious relationship with Pedersen commitments, we will think of multi-exponentiations $\vec{g}^{\vec{a}}$ and $\vec{h}^{\vec{b}}$ as commitments with randomness set equal to zero, and to $\vec{a},\vec{b}$ as openings with respect to commitment keys $\vec{g},\vec{h}$.

5.5.1 Main Idea

We now describe the basic step in our argument. Consider the common input for both prover and verifier to be of the form $(\mathbb{G}, p, \vec{g}, A, \vec{h}, B, z, m)$ where m divides n, the length of the vectors. For arbitrary n one can always reduce to the case where $m|n$ by appending at most $m-1$ random group elements to $\vec{g}$ and $\vec{h}$.

We split the bases for the multi-exponentiations into m sets $\vec{g} = (\vec{g}_1, \ldots, \vec{g}_m)$ and $\vec{h} = (\vec{h}_1, \ldots, \vec{h}_m)$, where each set has size $\frac{n}{m}$. We want to prove knowledge of vectors $\vec{a} = (\vec{a}_1, \ldots, \vec{a}_m)$ and $\vec{b} = (\vec{b}_1, \ldots, \vec{b}_m)$ such that

$$A = \vec{g}^{\vec{a}} = \prod_{i=1}^{m} \vec{g}_i^{\vec{a}_i} \qquad B = \vec{h}^{\vec{b}} = \prod_{i=1}^{m} \vec{h}_i^{\vec{b}_i} \qquad \vec{a} \cdot \vec{b} = \sum_{i=1}^{m} \vec{a}_i \cdot \vec{b}_i = z$$

The key idea is for the prover to replace A with A', a commitment to a shorter vector $\vec{a}' = \sum_{i=1}^{m} \vec{a}_i x^i$, given a random challenge $x \leftarrow \mathbb{Z}_p^*$ provided by the verifier. In the argument, the prover first computes and sends

$$A_k = \prod_{i=\max(1,1-k)}^{\min(m,m-k)} \vec{g}_i^{\vec{a}_{i+k}} \quad \text{for } k = 1-m, \dots, m-1$$

corresponding to the products over the diagonals of the following matrix

$$\begin{array}{c|cccc|c}
 & \vec{a}_1 & \vec{a}_2 & \cdots & \vec{a}_m & \\
\vec{g}_1 & \vec{g}_1^{\vec{a}_1} & \vec{g}_1^{\vec{a}_2} & \cdots & \vec{g}_1^{\vec{a}_m} & \\
\vdots & \ddots & \vec{g}_2^{\vec{a}_2} & \ddots & \vdots & \\
\vec{g}_{m-1} & \vec{g}_{m-1}^{\vec{a}_1} & \ddots & \ddots & \vec{g}_{m-1}^{\vec{a}_m} & A_{m-1} \\
\vec{g}_m & \vec{g}_m^{\vec{a}_1} & \vec{g}_m^{\vec{a}_2} & \cdots & \vec{g}_m^{\vec{a}_m} & \vdots \\
 & & & & & A_{m-2} \\
 & A_{1-m} & A_{2-m} & \cdots & & A_0 = A
\end{array}$$

Notice that $A_0 = A$ is already known to the verifier since it is part of the statement. The verifier now sends a random challenge $x \leftarrow \mathbb{Z}_p^*$.

At this point, both the prover and the verifier can compute $\vec{g}' := \prod_{i=1}^{m} \vec{g}_i^{x^{-i}}$ and $A' := \prod_{k=1-m}^{m-1} A_k^{x^k}$. If the prover is honest then we have $A' = (\vec{g}')^{\vec{a}'}$, namely A' is a commitment to $\vec{a}'$ under the key $\vec{g}'$. Furthermore, even if the prover is dishonest, we can show that if the prover can open A' with respect to the key $\vec{g}'$ for $2m-1$ different challenges, then we can extract opening $(\vec{a}_1, \dots, \vec{a}_m)$ corresponding to $A = \prod_{i=1}^{m} \vec{g}_i^{\vec{a}_i}$.

The same type of argument can be applied in parallel to B with the inverse challenge x^{-1} giving us a sum of the form $\vec{b}' = \sum_{i=1}^{m} \vec{b}_i x^{-i}$ and a new base $\vec{h}' = \prod_{i=1}^{m} \vec{h}_i^{x^i}$.

All that remains is to demonstrate that z is the constant term in the product $\vec{a}' \cdot \vec{b}' = \sum_{i=1}^{m} \vec{a}_i x^i \cdot \sum_{j=1}^{m} \vec{b}_j x^{-j}$. Similarly to A and B, the prover sends values

$$z_k = \sum_{i=\max(1,1-k)}^{\min(m,m-k)} \vec{a}_i \cdot \vec{b}_{i+k} \quad \text{for } k = 1-m, \dots, m-1$$

where $z_0 = z = \sum_{i=1}^{m} \vec{a}_i \cdot \vec{b}_i$, and shows that $z' := \vec{a}' \cdot \vec{b}' = \sum_{k=1-m}^{m-1} z_k x^{-k}$.

To summarise, after the challenge x has been sent, both parties compute $\vec{g}', A', \vec{h}', B', z'$ and then run an argument for the knowledge of $\vec{a}', \vec{b}'$ of length $\frac{n}{m}$. Given $n = m_\mu m_{\mu-1} \cdots m_1$, we recursively apply this reduction over the factors of n to obtain, after $\mu - 1$ iterations, vectors of length m_1. The prover concludes the argument by revealing a short witness associated with the last statement.

5.5.2 Formal description

We now give a formal description of the argument of knowledge introduced above.

Common input: $(\mathbb{G}, p, \vec{g}, A, \vec{h}, B, z, m_\mu = m, m_{\mu-1} = m', \ldots, m_1)$ such that $\vec{g}, \vec{h} \in \mathbb{G}^n$, $A, B \in \mathbb{G}$ and $n = \prod_{i=1}^{\mu} m_i$.

Prover's witness: $(\vec{a}_1, \ldots, \vec{a}_m, \vec{b}_1, \ldots, \vec{b}_m)$ satisfying

$$A = \prod_{i=1}^{m} \vec{g}_i^{\vec{a}_i} \qquad B = \prod_{i=1}^{m} \vec{h}_i^{\vec{b}_i} \qquad \sum_{i=1}^{m} \vec{a}_i \cdot \vec{b}_i = z$$

Argument if $\mu = 1$:

P → V: Send $(a_1, \ldots, a_m, b_1, , \ldots, b_m)$.

P ← V: Accept if and only if

$$A = \prod_{i=1}^{m} g_i^{a_i} \qquad B = \prod_{i=1}^{m} h_i^{b_i} \qquad \sum_{i=1}^{m} a_i b_i = z$$

Reduction if $\mu \neq 1$:

P → V: Send $A_{1-m}, B_{1-m}, z_{1-m}, \ldots, A_{m-1}, B_{m-1}, z_{m-1}$ where

$$A_k = \prod_{i=\max(1,1-k)}^{\min(m,m-k)} \vec{g}_i^{\vec{a}_{i+k}} \qquad B_k = \prod_{i=\max(1,1-k)}^{\min(m,m-k)} \vec{h}_i^{\vec{b}_{i+k}} \qquad z_k = \sum_{i=\max(1,1-k)}^{\min(m,m-k)} \vec{a}_i \cdot \vec{b}_{i+k}$$

Observe $A_0 = A, B_0 = B, z_0 = z$ so they can be omitted from the message.

P ← V: $x \leftarrow \mathbb{Z}_p^*$.

Both prover and verifier compute a reduced statement of the form

$$(\mathbb{G}, p, \vec{g}', A', \vec{h}', B', z', m_{\mu-1}, \ldots, m_1)$$

where

$$\vec{g}' = (\vec{g}'_1, \ldots, \vec{g}'_{m'}) = \prod_{i=1}^{m} \vec{g}_i^{x^{-i}} \qquad A' = \prod_{k=1-m}^{m-1} A_k^{x^k}$$
$$\vec{h}' = (\vec{h}'_1, \ldots, \vec{h}'_{m'}) = \prod_{i=1}^{m} \vec{h}_i^{x^{i}} \qquad B' = \prod_{k=1-m}^{m-1} B_k^{x^{-k}} \qquad z' = \sum_{k=1-m}^{m-1} z_k x^{-k}$$

The prover computes a new witness as $(\vec{a}'_1, \ldots, \vec{a}'_{m'}) = \sum_{i=1}^{m} \vec{a}_i x^i$ and $(\vec{b}'_1, \ldots, \vec{b}'_{m'}) = \sum_{i=1}^{m} \vec{b}_i x^{-i}$.

5.5.3 Security Analysis.

Theorem 7. *The argument has perfect completeness and statistical witness extended emulation for either extracting a non-trivial discrete logarithm relation or a valid witness.*

Proof. Perfect completeness can be verified directly. To prove witness-extended emulation we start by giving an extractor that either extracts a witness for the original statement or a non-trivial discrete logarithm relation.

For $\mu = 1$ we have (perfect) witness-extended emulation since the prover reveals a witness and the verifier checks it.

Before discussing extraction in the recursive step, note that if we get a non-trivial discrete logarithm relation for $\vec{g}'_1, \ldots, \vec{g}'_{m'}$ then we also get a non-trivial discrete logarithm relation for $\vec{g}_1, \ldots, \vec{g}_m$, since $x \neq 0$. A similar argument applies to $\vec{h}'_1, \ldots, \vec{h}'_{m'}$ and $\vec{h}_1, \ldots, \vec{h}_m$.

Now, assume we get witness $\vec{a}', \vec{b}'$ such that

$$A' = \prod_{k=1-m}^{m-1} A_k^{x^k} = \left(\prod_{i=1}^{m} \vec{g}_i^{x^{-i}}\right)^{\vec{a}'} \quad B' = \prod_{k=1-m}^{m-1} B_k^{x^{-k}} = \left(\prod_{i=1}^{m} \vec{h}_i^{x^{i}}\right)^{\vec{b}'} \quad \vec{a}' \cdot \vec{b}' = \sum_{k=1-m}^{m-1} z_k x^{-k}$$

for $2m-1$ different challenges $x \in \mathbb{Z}_p^*$. We will show that they yield either a witness for the original statement, or a non-trivial discrete logarithm relation for either $\vec{g}_1, \ldots, \vec{g}_m$ or $\vec{h}_1, \ldots, \vec{h}_m$.

Take $2m-1$ different challenges $x \in \mathbb{Z}_p^*$. They form a shifted Vandermonde matrix with rows $(x^{1-m}, x^{2-m}, \ldots, x^{m-1})$. By taking appropriate linear combinations of the vectors we can obtain any unit vector $(0, \ldots, 0, 1, 0, \ldots, 0)$. Taking the same linear combinations of the $2m-1$ equations

$$\prod_{k=1-m}^{m-1} A_k^{x^k} = \left(\prod_{i=1}^{m} \vec{g}_i^{x^{-i}}\right)^{\vec{a}'} \qquad \text{we get vectors } \vec{a}_{k,i} \text{ such that} \qquad A_k = \prod_{i=1}^{m} \vec{g}_i^{\vec{a}_{k,i}}$$

For each of the $2m-1$ challenges, we now have $\prod_{k=1-m}^{m-1} A_k^{x^k} = \left(\prod_{i=1}^{m} \vec{g}_i^{\,x^{-i}}\right)^{\vec{a}'}$, which means that *for all i* we have

$$x^{-i}\vec{a}' = \sum_{k=1-m}^{m-1} \vec{a}_{k,i} x^k$$

unless we encounter a non-trivial discrete logarithm relation for $\vec{g}_1,\ldots,\vec{g}_m$. This means that $\vec{a}' = \sum_{k=1-m}^{m-1} \vec{a}_{k,i} x^{k+i}$ for all i, and in particular $\sum_{k=1-m}^{m-1} \vec{a}_{k,i} x^{k+i} = \sum_{k=1-m}^{m-1} \vec{a}_{k,1} x^{k+1} = \sum_{k=1-m}^{m-1} \vec{a}_{k,m} x^{k+m}$. Matching terms of degree outside $\{1,\ldots,m\}$ reveals $\vec{a}_{k,i} = 0$ for $k+i \notin \{1,\ldots,m\}$. Defining $\vec{a}_i = \vec{a}_{0,i}$, and matching terms of similar degree we get

$$\vec{a}_{k,i} = \begin{cases} \vec{a}_{k+i} & \text{if } k+i \in \{1,\ldots,m\} \\ 0 & \text{otherwise} \end{cases}$$

This means

$$\vec{a}' = \sum_{k=1-m}^{m-1} \vec{a}_{k,1} x^{k+1} = \sum_{k=0}^{m-1} \vec{a}_{k+1} x^{k+1} = \sum_{i=1}^{m} \vec{a}_i x^i$$

A similar analysis of $B_{1-m},\ldots,B_{m-1}$ and openings $\vec{b}'$ for $2m-1$ different challenges $x^{-1} \in \mathbb{Z}_p^*$ gives us either a non-trivial discrete logarithm relation for $\vec{h}_1,\ldots,\vec{h}_m$ or vectors $\vec{b}_i$ such that $\vec{b}' = \sum_{i=1}^{m} \vec{b}_i x^{-i}$ and $B = \prod_{i=1}^{m} \vec{h}_i^{\vec{b}_i}$.

Finally, with $\sum_{i=1}^{m} \vec{a}_i x^i \cdot \sum_{j=1}^{m} \vec{b}_j x^{-j} = \sum_{k=1-m}^{m-1} z_k x^{-k}$ for $2m-1$ different challenges we get $z = z_0 = \sum_{i=1}^{m} \vec{a}_i \cdot \vec{b}_i$.

We can now apply the forking lemma to a tree of size $(2m_\mu - 1)(2m_{\mu-1} - 1)\cdots(2m_2 - 1) \leq n^2$, which is polynomial in λ, to conclude that the argument has witness-extended emulation. □

Note that the argument is not zero-knowledge, as it is not necessary for its use in the next section.

5.5.4 Efficiency.

The recursive argument uses $2\mu - 1$ moves. The communication cost of all steps sums up to $4\sum_{i=2}^{\mu}(m_i - 1)$ group elements and $2\sum_{i=2}^{\mu}(m_i - 1) + 2m_1$ field elements.

At each iteration, the main cost for the prover is computing the A_k and B_k values, using less than $\frac{4(m_\mu^2 m_{\mu-1} \ldots m_1)}{\log(m_\mu \ldots m_1)}$ group exponentiations via multi-exponentiation techniques, and the z_k values using $m_\mu^2 m_{\mu-1} \cdots m_1$ field multiplications. The cost of computing the reduced statements is dominated by $\frac{2(m_\mu m_{\mu-1} \ldots m_1)}{\log m_\mu}$ group exponentiations for both the prover and the verifier. In the case where $m_\mu = \ldots = m_1 = m$, the verifier complexity is bounded above by $\frac{2m^{\mu}}{\log m}\frac{m}{m-1}$ group exponentiations. The prover complexity is bounded above by $\frac{6m^{\mu+1}}{\log m}\frac{m}{m-1}$ group exponentiations and $m^{\mu+1}\frac{m}{m-1}$ field multiplications.

5.6 Logarithmic Communication Argument for Arithmetic Circuit Satisfiability

In this section, we revisit zero knowledge arguments for the satisfiability of an arithmetic circuit under the discrete logarithm assumption. We will explain how to build an argument with square root communication complexity, and superior efficiency to the argument of [Gro09b]. We then observe that our new argument involves computing a large inner product, and can achieve as good as logarithmic communication complexity by using our recursive inner product argument.

At a high level, we transform an arithmetic circuit into two kinds of equations. Multiplication gates are directly represented as equations of the form $a \cdot b = c$, where a, b, c represent the left, right and output wires. We will arrange these values in matrix form producing a Hadamard matrix product. This process will lead to duplicate values, when a wire is the output of one multiplication gate and the input of another, or when it is used as input multiple times. We keep track of this by using a series of linear constraints. For example, if the output of the first multiplication gate is the right input of the second, we would write $c_1 - b_2 = 0$.

We also add linear constraints representing the addition and multiplication by constant gates of the circuit. We then rewrite those equations so that the only wires that are referenced in the equations are those linked to (non-constant) multiplication gates. We describe this process in Section 5.3.1.

Finally, we fold both the Hadamard matrix product and the linear constraints into a single polynomial equation, where a Laurent polynomial has 0 as its constant term, and use the construction of Section 5.4 to prove this. We can optionally integrate the inner product argument of Section 5.5 to reduce communication.

Our technique improves on the efficiency of [Gro09b] by making three main changes, each resulting in efficiency improvements.

1. We do not need commitments to the input and output wires of addition gates. We handle addition gates with linear consistency equations thus yielding a significant performance improvement proportional to the number of addition gates. This par-

allels [GGPR13] who also manage to eliminate addition gates when constructing Quadratic Arithmetic Programs from circuits.

2. We avoid black-box reductions to zero-knowledge arguments for generic linear algebra statements and instead design an argument directly for arithmetic circuit satisfiability. As a result, our square-root argument has only 5 moves, while the argument from [Gro09b] requires 7 moves. We note that [Seo11] reduced the complexity of [Gro09b] to 5 moves as well, but at a significant computational overhead whereas we also reduce the computational cost.
3. We use our protocol from Section 5.4 to reduce the communication costs of a polynomial commitment.

These improvements give us a square root communication complexity with respect to the number of multiplication gates in the circuit. This is because for a circuit with $N = mn$ multiplication gates, the prover makes $3m$ commitments to wire values in his first move, and later provides an opening consisting of n field elements to a homomorphic combination of these commitments. Optimising the parameters by choosing $m \approx n \approx \sqrt{N}$ leads to square root complexity.

In our square root complexity argument, the verifier uses the n field elements to check an inner product relation. Our key idea to reduce communication further is to use our inner product evaluation argument instead of sending these field elements. This allows for verification of the inner product, and also provides an argument of knowledge of the opening of the commitment. We no longer need to open a large commitment, leading to a drastic reduction in communication complexity depending on the settings for the inner product argument.

Below we give a first informal exposition of our arguments, and follow with a formal description.

5.6.1 Reduction of Circuit Satisfiability to a Hadamard Matrix Product and Linear Constraints.

We consider an arithmetic circuit containing $N = mn$ multiplication gates over a field $\mathbb{Z}_p$. Without loss of generality, we assume that the circuit has been pre-processed (see Appendix 5.3.1 for a way to do this) , so that the input and the output wires feed into and go out from multiplication gates only. We number the multiplication gates from 1 to N and we arrange the inputs and outputs of these gates into three $m \times n$ matrices A, B and C such that the (i,j) entries of the matrices correspond to the left input, right input and output of the same multiplication gate.

As shown in Sect. 5.3.1, anarithmetic circuit can be described as a system of equations in the entries of the above matrices. The multiplication gates define a set of N equations

$$A \circ B = C \tag{5.1}$$

where $\circ$ is the Hadamard (entry-wise) product. The circuit description also contains constraints on the wires between multiplication gates. Denoting the rows of the matrices A, B, C as

$$\vec{a}_i = (a_{i,1}, \dots, a_{i,n}) \qquad \vec{b}_i = (b_{i,1}, \dots, b_{i,n}) \qquad \vec{c}_i = (c_{i,1}, \dots, c_{i,n}) \qquad \text{for } i \in \{1, \dots, m\}$$

these constraints can be expressed as $Q < 2N$ linear equations of inputs and outputs of multiplication gates of the form

$$\sum_{i=1}^{m} \vec{a}_i \cdot \vec{w}_{q,a,i} + \sum_{i=1}^{m} \vec{b}_i \cdot \vec{w}_{q,b,i} + \sum_{i=1}^{m} \vec{c}_i \cdot \vec{w}_{q,c,i} = K_q \qquad \text{for } q \in \{1, \dots, Q\} \tag{5.2}$$

for constant vectors $\vec{w}_{q,a,i}, \vec{w}_{q,b,i}, \vec{w}_{q,c,i}$ and scalars K_q.

For example, suppose that the circuit contains a single addition gate, with $a_{1,1}$ and $a_{1,2}$ as inputs, and $b_{1,1}$ as output. In this case, $Q = 1$ and we would set $\vec{w}_{1,a,1} = (1, 1, 0, \dots, 0)$, $\vec{w}_{1,b,1} = (-1, 0, \dots, 0)$, and all other $\vec{w}$ vectors would be set to $\vec{0}$. Then (5.2) would simply read

$$a_{1,1} + a_{1,2} - b_{1,1} = 0$$

to capture the constraint imposed by the addition gate.

In total, to capture all multiplications and linear constraints, we have $N + Q$ equations that the wires must satisfy in order for the circuit to be satisfiable.

5.6.2 Reduction to a Single Polynomial Equation.

Let Y be a formal indeterminate. We will reduce the $N + Q$ equations above to a single polynomial equation in Y by embedding each equation into a distinct power of Y. In our argument we will then require the prover to prove that this single equation holds when replacing Y by a random challenge received from the verifier.

Let $\vec{Y}'$ denote the vector $(Y^m, \dots, Y^{mn})$ and $\vec{Y}$ denote $(Y, Y^2, \dots, Y^m)$. Then, we can multiply (5.1) by $\vec{Y}$ from the left and $\vec{Y}'^T$ on the right to obtain $\vec{Y}(A \circ B)\vec{Y}'^T = \vec{Y}C\vec{Y}'^T$, or equivalently

$$\sum_{i=1}^{m} Y^i(\vec{a}_i \circ \vec{b}_i) \cdot \vec{Y}' = \sum_{i=1}^{m} Y^i(\vec{c}_i \cdot \vec{Y}')$$

Since $(\vec{a} \circ \vec{b}) \cdot \vec{Y}' = \vec{a} \cdot (\vec{b} \circ \vec{Y}')$, we obtain the following expression

$$\sum_{i=1}^{m} \vec{a}_i \cdot (\vec{b}_i \circ \vec{Y}')Y^i = \left(\sum_{i=1}^{m} \vec{c}_i Y^i \cdot \vec{Y}'\right)$$

This is easily seen to be equivalent to (5.1), because $a_{i,j}b_{i,j} = c_{i,j}$ appears in the coefficients of Y^{i+jm}, and $i + jm$ takes every value from $m+1$ to $M = N+m$ exactly once.

Moreover, the Q linear constraints on the wires in Eq. 5.2 are satisfied if and only if

$$\sum_{q=1}^{Q}\left(\sum_{i=1}^{m} \vec{a}_i \cdot \vec{w}_{q,a,i} + \sum_{i=1}^{m} \vec{b}_i \cdot \vec{w}_{q,b,i} + \sum_{i=1}^{m} \vec{c}_i \cdot \vec{w}_{q,c,i}\right) Y^q = \sum_{q=1}^{Q} K_q Y^q$$

since the qth constraint arises from comparing the coefficients of Y^q. Combining the two polynomial equations by adding them after multiplying the latter by Y^M, and swapping summations, we see that the circuit is satisfied if and only if

$$\left(\sum_{i=1}^{m} \vec{a}_i \cdot (\vec{b}_i \circ \vec{Y}')Y^i\right) + \sum_{i=1}^{m} \vec{a}_i \cdot \left(\sum_{q=1}^{Q} \vec{w}_{q,a,i} Y^{M+q}\right) + \sum_{i=1}^{m} \vec{b}_i \cdot \left(\sum_{q=1}^{Q} \vec{w}_{q,b,i} Y^{M+q}\right)$$
$$+ \sum_{i=1}^{m} \vec{c}_i \cdot \left(-Y^i\vec{Y}' + \sum_{q=1}^{Q} \vec{w}_{q,c,i} Y^{M+q}\right) = \left(\sum_{q=1}^{Q} K_q Y^{M+q}\right)$$

Let us define

$$\vec{w}_{a,i}(Y) = \sum_{q=1}^{Q} \vec{w}_{q,a,i} Y^{M+q} \qquad \vec{w}_{b,i}(Y) = \sum_{q=1}^{Q} \vec{w}_{q,b,i} Y^{M+q}$$
$$\vec{w}_{c,i}(Y) = -Y^i\vec{Y}' + \sum_{q=1}^{Q} \vec{w}_{q,c,i} Y^{M+q} \qquad K(Y) = \sum_{q=1}^{Q} K_q Y^{M+q}$$

Then the circuit is satisfied if and only if

$$\sum_{i=1}^{m} \vec{a}_i \cdot (\vec{b}_i \circ \vec{Y}')Y^i + \sum_{i=1}^{m} \vec{a}_i \cdot \vec{w}_{a,i}(Y) + \sum_{i=1}^{m} \vec{b}_i \cdot \vec{w}_{b,i}(Y) + \sum_{i=1}^{m} \vec{c}_i \cdot \vec{w}_{c,i}(Y) - K(Y) = 0 \qquad (5.3)$$

In the argument, the prover will commit to $\vec{a}_i, \vec{b}_i$ and $\vec{c}_i$. The verifier will then issue a random challenge $y \leftarrow \mathbb{Z}_p^*$ and the prover will convince the verifier that the committed values satisfy Eq. 5.3, evaluated on y. If the committed values do not satisfy the polynomial equation, the probability the equality holds for a random y is negligible, so the prover is unlikely to be able to convince the verifier.

5.6.3 Square Root Communication Argument

In order to show that (5.3) is satisfied, we craft a special Laurent polynomial $t(X)$ in a second formal indeterminate X, whose constant coefficient is exactly twice the left-hand side of (5.3). Therefore, this polynomial will have zero constant term if and only if (5.3) is satisfied. In our argument this is proved using the polynomial commitment protocol of Section 5.4. We define

$$\begin{aligned}
\vec{r}(X) &:= \sum_{i=1}^{m} \vec{a}_i y^i X^i + \sum_{i=1}^{m} \vec{b}_i X^{-i} + X^m \sum_{i=1}^{m} \vec{c}_i X^i + \vec{d} X^{2m+1} \\
\vec{s}(X) &:= \sum_{i=1}^{m} \vec{w}_{a,i}(y) y^{-i} X^{-i} + \sum_{i=1}^{m} \vec{w}_{b,i}(y) X^i + X^{-m} \sum_{i=1}^{m} \vec{w}_{c,i}(y) X^{-i} \\
\vec{r}'(X) &:= \vec{r}(X) \circ \vec{y}' + 2\vec{s}(X) \\
t(X) &:= \vec{r}(X) \cdot \vec{r}'(X) - 2K(y)
\end{aligned}$$

Here $\vec{y}'$ is the vector $\vec{Y}'$ evaluated at y, and $\vec{d}$ is a blinding vector consisting of random scalars that the prover commits to in the first round. In the square root argument the prover will reveal $\vec{r}(x)$ for a randomly chosen challenge $x \in \mathbb{Z}_p^*$, and the blinding vector $\vec{d}$ ensures that we can reveal $\vec{r}(x)$ without leaking information about $\vec{a}_i, \vec{b}_i$ and $\vec{c}_i$. We also observe that $\vec{s}(x)$ is efficiently computable from public information about the circuit and the challenges.

We have designed these polynomials such that the constant term of $\vec{r} \cdot (\vec{r} \circ \vec{y}')$ is equal to $2\sum_{i=1}^{m} \vec{a}_i \cdot (\vec{b}_i \circ \vec{y}')y^i$ and the constant term of $\vec{r} \cdot \vec{s}$ is equal to $\sum_{i=1}^{m} \vec{a}_i \cdot \vec{w}_{a,i}(y) + \sum_{i=1}^{m} \vec{b}_i \cdot \vec{w}_{b,i}(y) + \sum_{i=1}^{m} \vec{c}_i \cdot \vec{w}_{c,i}(y)$. We conclude that the constant term of $t(X)$ is exactly twice the left-hand side of (5.3), and is therefore zero if and only if the circuit is satisfied.

We are now in a position to describe an argument with square root communication complexity.

The prover first commits to vectors $\vec{a}_i, \vec{b}_i, \vec{c}_i$ and $\vec{d}$ and the verifier replies with a challenge $y \leftarrow \mathbb{Z}_p^*$. The prover computes $t(X)$ and commits to it by using the algorithm PolyCommit defined in Section 5.4. Then, the verifier sends a random challenge $x \leftarrow \mathbb{Z}_p^*$ and the prover responds by revealing $\vec{r}(x)$ and blinded openings pe of $t(X)$ obtained by running algorithm PolyEval as described in Section 5.4.

The verifier first checks that $\vec{r}(x)$ is consistent with the previously sent commitments of $\vec{a}_i, \vec{b}_i, \vec{c}_i$ and $\vec{d}$ using the homomorphic properties of the commitment scheme. She also computes $\vec{s}(x), \vec{r}'(x)$ and K. Then, she computes $v = t(x)$ using the PolyVerify algorithm of Section 5.4, and checks if $v = \vec{r}(x) \cdot \vec{r}'(x) - 2K$. The verifier accepts the argument if both checks are satisfied.

As described so far, the argument requires communicating $O(m)$ group elements and $O(n)$ field elements, so setting $m \approx n$ leads to square root communication. The argument improves on [Gro09b, Seo11] by requiring only 5 moves without computational overhead and significantly reduces the computational complexity. However, breaking this ostensible square root communication barrier (Sect. 2.1.2) requires new ideas that we describe in the next section.

5.6.4 Breaking the Square Root Barrier

The square root complexity argument described above was designed so that the verifier uses $\vec{r} = \vec{r}(x)$ to check the inner product $v = \vec{r} \cdot \vec{r}' - 2K$, where v is the evaluation of a committed polynomial at x. Sending $\vec{r}$ has a cost of n field elements. In order to break the square root barrier we try to avoid sending $\vec{r}$ directly so that we can then let n be larger and m be smaller and thus globally lower the communication of the argument.

Rather than sending $\vec{r}$ to the verifier, the prover could instead send commitments to $\vec{r}$ and $\vec{r}'$, and use our inner product argument to show that $v + 2K$ was a correctly formed inner product. In fact, the prover does not even need to send commitments to $\vec{r}$ and $\vec{r}'$! The verifier can compute a commitment to $\vec{r}(x)$ directly from A_i, B_i, C_i and D, the commitments to $\vec{a}_i, \vec{b}_i$, $\vec{c}_i$ and $\vec{d}$ which were previously used to check that $\vec{r}$ is correctly formed

$$\mathrm{Com}_{ck}(\vec{r};0) = \mathrm{Com}_{ck}(0;-\rho)\left[\prod_{i=1}^{m} A_i^{x^i y^i}\right]\left[\prod_{i=1}^{m} B_i^{x^{-i}}\right]\left[\prod_{i=1}^{m} C_i^{x^{m+i}}\right] D^{x^{2m+1}} = \vec{g}^{\vec{r}}$$

where ρ is an appropriate randomness value, which is sent by the prover to the verifier, and the vector $\vec{g} = (g_1, \ldots, g_n)$ for a given commitment key $ck = (\mathbb{G}, p, g, g_1, \ldots, g_n)$.

As for a commitment to $\vec{r}'$, we observe that the Pedersen commitment, besides its well-known homomorphic properties with respect to the message and the randomness, also has the useful property that it is homomorphic *with respect to the commitment key*. Specifically, let $\vec{h} = (g_1^{y^{-m}}, \ldots, g_n^{y^{-mn}})$, so that $\vec{g}^{\vec{r}} = \vec{h}^{\vec{r} \circ \vec{y}'}$. Multiplying $\vec{g}^{\vec{r}}$ by $\vec{h}^{2\vec{s}}$, the verifier obtains $\mathrm{Com}_{ck'}(\vec{r}';0) = \vec{h}^{\vec{r}'}$, with respect to the new commitment key ck' which uses $\vec{h}$ instead of $\vec{g}$.

We note that $\vec{h}$ and $\vec{s} = \vec{s}(x)$ can be computed by the verifier.

Now the prover and verifier can run the inner product argument with statement

$$(\mathbb{G}, p, \vec{g}, r, \vec{h}, r', v + 2K, m_\mu, m_{\mu-1}, \ldots, m_1) \qquad \text{where}$$

$$\begin{array}{llll}
ck & = (\mathbb{G}, p, g, \vec{g}) & n = m_\mu m_{\mu-1} \cdots m_1 & \\
\vec{g} & = (g_1, g_2, \ldots, g_n) & \vec{h} = (g_1^{y^{-m}}, g_2^{y^{-2m}}, \ldots, g_n^{y^{-mn}}) & \\
R & = \mathrm{Com}_{ck}(0; -\rho) \left[\prod_{i=1}^m A_i^{x^i y^i}\right] & \left[\prod_{i=1}^m B_i^{x^{-i}}\right] \left[\prod_{i=1}^m C_i^{x^{m+i}}\right] D^{x^{2m+1}} = \vec{g}^{\vec{r}} & \\
R' & = R \cdot \vec{h}^{2\vec{s}} = \vec{h}^{\vec{r}'} & &
\end{array}$$

and the prover's witness is $\vec{r}, \vec{r'}$.

The values of $m_\mu, \ldots, m_1$ can be chosen according to the desired efficiency of the circuit satisfiability argument.

5.6.5 Formal Description

We now give the formal description of the above arguments of knowledge for the satisfiability of an arithmetic circuit C. Both prover and verifier take the move parameter μ as common input. For square root communication complexity, the inner product argument is not used and we set $\mu = 0$. For $\mu > 0$, the common input includes the values $(m_\mu, \ldots, m_1)$ used in the inner product argument. The description of the arithmetic circuit C is given as a number N of multiplication gates and the values $\vec{w}_{q,a,i}, \vec{w}_{q,b,i}, \vec{w}_{q,c,i}$, which specify linear consistency constraints between the input and output values of the multiplication gates.

Common Input: $(ck, C, N, m, n, m_1', m_2', n', m_\mu, \ldots, m_1, \mu)$ where ck is a commitment key, C is the description of an arithmetic circuit with $N = mn$ multiplication gates, μ is the move parameter and $n = m_\mu \cdots m_1$. Parameters (m_1', m_2', n') are set to satisfy both $3m \leq m_1' n'$ and $4m + 2 \leq m_2' n'$.

Prover's Witness: Satisfying assignments $\vec{a}_i, \vec{b}_i$ and $\vec{c}_i$ to the wires of C.

Argument:

P → V: Pick randomness $\alpha_1, \beta_1, \gamma_1, \ldots, \alpha_m, \beta_m, \gamma_m, \delta \leftarrow \mathbb{Z}_p$ and blinding vector $\vec{d} \leftarrow \mathbb{Z}_p^n$. Compute for $i \in \{1, \ldots, m\}$

$$A_i = \mathrm{Com}(\vec{a}_i; \alpha_i) \quad B_i = \mathrm{Com}(\vec{b}_i; \beta_i) \quad C_i = \mathrm{Com}(\vec{c}_i; \gamma_i) \quad D = \mathrm{Com}(\vec{d}; \delta).$$

Send to the verifier $A_1, B_1, C_1, \ldots, A_m, B_m, C_m, D$.

P ← V: $y \leftarrow \mathbb{Z}_p^*$.

As argued before, the circuit determines vectors of polynomials $\vec{w}_{a,i}(Y)$, $\vec{w}_{b,i}(Y)$, $\vec{w}_{c,i}(Y)$ and $K(Y)$ such that C is satisfiable if and only if

$$\sum_{i=1}^{m} \vec{a}_i \cdot (\vec{b}_i^T \circ \vec{Y}')Y^i + \sum_{i=1}^{m} \vec{a}_i \cdot \vec{w}_{a,i}(Y) + \sum_{i=1}^{m} \vec{b}_i \cdot \vec{w}_{b,i}(Y) + \sum_{i=1}^{m} \vec{c}_i \cdot \vec{w}_{c,i}(Y) = K(Y)$$

where $\vec{Y}' = (Y^m, \ldots, Y^{mn})$. Given y, both the prover and verifier can compute $K = K(y)$, $\vec{w}_{a,i} = \vec{w}_{a,i}(y)$, $\vec{w}_{b,i} = \vec{w}_{b,i}(y)$ and $\vec{w}_{c,i} = \vec{w}_{c,i}(y)$.

P → V: Compute Laurent polynomials $\vec{r}, \vec{s}, \vec{r}'$, which have vector coefficients, and Laurent polynomial t, in the indeterminate X

$$\begin{aligned}
\vec{r}(X) &= \sum_{i=1}^{m} \vec{a}_i y^i X^i + \sum_{i=1}^{m} \vec{b}_i X^{-i} + X^m \sum_{i=1}^{m} \vec{c}_i X^i + \vec{d}X^{2m+1} \\
\vec{s}(X) &= \sum_{i=1}^{m} \vec{w}_{a,i} y^{-i} X^{-i} + \sum_{i=1}^{m} \vec{w}_{b,i} X^i + X^{-m} \sum_{i=1}^{m} \vec{w}_{c,i} X^{-i} \\
\vec{r}'(X) &= \vec{r}(X) \circ \vec{y}' + 2\vec{s}(X) \\
t(X) &= \vec{r}(X) \cdot \vec{r}'(X) - 2K = \sum_{k=-3m}^{4m+2} t_k X^k
\end{aligned}$$

When the wires $\vec{a}_i, \vec{b}_i, \vec{c}_i$ correspond to a satisfying assignment, the Laurent polynomial $t(X)$ will have constant term $t_0 = 0$.

Commit to $t(X)$ by running

$$(\mathsf{pc}, \mathsf{st}) \leftarrow \text{PolyCommit}(ck, m_1', m_2', n', t(X))$$

Send pc to the verifier.

P ← V: $x \leftarrow \mathbb{Z}_p^*$

P → V: Compute $\text{PolyEval}(\mathsf{st}, x) \rightarrow \mathsf{pe}$, and

$$\vec{r} = \sum_{i=1}^{m} \vec{a}_i x^i y^i + \sum_{i=1}^{m} \vec{b}_i x^{-i} + x^m \sum_{i=1}^{m} \vec{c}_i x^i + \vec{d} x^{2m+1}$$

$$\rho = \sum_{i=1}^{m} \alpha_i x^i y^i + \sum_{i=1}^{m} \beta_i x^{-i} + x^m \sum_{i=1}^{m} \gamma_i x^i + \delta x^{2m+1}$$

- **If $\mu = 0$** : the inner product argument is not used. The prover sends $(\mathsf{pe}, \vec{r}, \rho)$ to the verifier.
- **If $\mu > 0$** : the inner product argument is used. The prover computes $\vec{r}' = \vec{r}'(x)$ and sends (pe, ρ) to the verifier.

Verification: First, the verifier computes

$$\text{PolyVerify}(ck, m_1', m_2', n', \mathsf{pc}, \mathsf{pe}, x) \rightarrow v$$

and rejects the argument if $v = \perp$.

- **If $\mu = 0$** : the inner product argument is not used. The verifier computes $\vec{r}' = \vec{r} \circ \vec{y}' + 2\vec{s}(x)$, and accepts only if

$$\begin{aligned} \vec{r} \cdot \vec{r}' - 2K &= v \\ \text{Com}_{ck}(\vec{r}; \rho) &= \left[\prod_{i=1}^{m} A_i^{x^i y^i}\right] \left[\prod_{i=1}^{m} B_i^{x^{-i}}\right] \left[\prod_{i=1}^{m} C_i^{x^{m+i}}\right] D^{x^{2m+1}} \end{aligned}$$

- **If $\mu > 0$** : prover and verifier run the inner product argument with common input

$$(\mathbb{G}, p, \vec{g}, R, \vec{h}, R', v + 2K, m_\mu, m_{\mu-1}, \ldots, m_1) \qquad \text{where}$$

$$\begin{aligned} ck &= (\mathbb{G}, p, g, \vec{g}) & n &= m_\mu m_{\mu-1} \cdots m_1 \\ \vec{g} &= (g_1, g_2, \ldots, g_n) & \vec{h} &= (g_1^{y^{-m}}, g_2^{y^{-2m}}, \ldots, g_n^{y^{-mn}}) \\ R &= \text{Com}_{ck}(0; -\rho) \left[\prod_{i=1}^{m} A_i^{x^i y^i}\right] & \left[\prod_{i=1}^{m} B_i^{x^{-i}}\right] & \left[\prod_{i=1}^{m} C_i^{x^{m+i}}\right] D^{x^{2m+1}} = \vec{g}^{\vec{r}} \\ R' &= R \cdot \vec{h}^{2\vec{s}(x)} = \vec{h}^{\vec{r}'} \end{aligned}$$

and the prover's witness is $\vec{r}$ and $\vec{r'}$.

The verifier accepts if the inner product argument is accepting.

5.6.6 Security Analysis.

Theorem 8. *The argument for satisfiability of an arithmetic circuit has perfect completeness, perfect special honest verifier zero-knowledge and statistical witness-extended emulation for extracting either a breach of the binding property of the commitment scheme or a witness for the satisfiability of the circuit.*

Proof. Perfect completeness follows by inspection and using the fact that the polynomial commitment protocol and inner product argument also have perfect completeness.

For perfect special honest verifier zero-knowledge we are given $y, x \in \mathbb{Z}_p^*$, which allows us to compute $\vec{w}_{a,i}, \vec{w}_{b,i}, \vec{w}_{c,i}$ and K from the circuit. The simulator picks $\vec{r} \leftarrow \mathbb{Z}_p^n$ and $\rho \leftarrow \mathbb{Z}_p$ and random commitments A_i, B_i and C_i. It computes

$$D = \left[\prod_{i=1}^{m} A_i^{x^i y^i} B_i^{x^{-i}} C_i^{x^{m+i}} \mathrm{Com}_{ck}(-\vec{r}; -\rho)\right]^{-x^{-2m-1}} \qquad v = \vec{r} \cdot \vec{r}' - 2K$$

and simulates pc and pe from the polynomial commitment protocol.

To see that the simulated components have the same distribution as a real argument observe that since the commitment scheme is perfectly hiding the commitments A_i, B_i and C_i have the same distribution as in a real argument. Also, in both the simulation and a real argument $\vec{r}$ and ρ are uniformly random. Given these values the commitment D is uniquely defined. Furthermore, since the polynomial commitment protocol is perfect special honest verifier zero-knowledge, pc and pe have the same distribution as in a real argument, conditioned on the value of v.

When $\mu > 0$ we simply remove $\vec{r}$ from the transcript and execute a fresh run of the inner product argument, given our knowledge of $\vec{r}$.

It remains to show that we have witness-extended emulation. We treat the cases $\mu = 0$ and $\mu > 0$ separately.

Square Root Argument. Assume that we have $N + Q$ different challenges $y \in \mathbb{Z}_p^*$ for the same initial message, and for each of these challenges a further $7m + 3$ different challenges $x \in \mathbb{Z}_p^*$ for the same third message, all with valid answers. We begin by showing that from this information we either extract a satisfying assignment to the wires $\vec{a}_i, \vec{b}_i, \vec{c}_i$ in the circuit, or encounter a breach of the binding property of the commitment scheme.

Let us first consider a fixed initial transcript $(A_1, \dots, A_m, \dots, C_m, D, y, \mathsf{pc})$ and suppose we have valid arguments with $3m + 2$ different values of x. Then the vectors $(x^{-m}, \dots, x^{2m+1})$ form the rows of a shifted Vandermonde matrix and we can obtain any unit vector $(0, \dots, 1, \dots, 0)$ by taking an appropriate linear combination of these vectors. By taking this linear combination of the $3m+2$ verification equations $D^{x^{2m+1}} \prod_{i=1}^{m} A_i^{x^i y^i} B_i^{x^{-i}} C_i^{x^{m+i}} = \mathrm{Com}_{ck}(\vec{r}; \rho)$, we can then extract openings to each A_i, B_i and C_i, since $y \in \mathbb{Z}_P^*$.

We have valid arguments for $(m_1' + m_2')n' + 1 = 7m + 3$ different challenges $x \in \mathbb{Z}_p^*$.

By the proof of Theorem 6 this implies that there exists a Laurent polynomial $t(X) = \sum_{i=-n'm'_1}^{n'm'_2} t_i x^i$ with constant term $t_0 = 0$ that is consistent with respect to all $3m+2$ evaluations of $\vec{r}(X) \cdot \vec{r}'(X) - 2K$. This directly implies that Equation 5.3 holds for $Y = y$.

Finally, suppose that this holds for $N+Q$ different challenges $y \in \mathbb{Z}_p^*$. Then, we have equality of polynomials in Equation 5.3, since a non-zero polynomial of degree $N+Q-1$ cannot have $N+Q$ roots. This means that the circuit is satisfied.

Now we can apply the forking lemma to get witness-extended emulation, as the tree formed by the transcripts has size $(N+Q) \cdot (7m+3)$ which is polynomial in λ.

Inner Product Variant Assume that we have $(N+Q) \cdot (7m+3) \cdot (2m_\mu - 1) \dots (2m_2 - 1)$ accepting transcripts for the same statement arranged in a tree as follows:

- The root is labeled with the statement.
- Each of the $(N+Q)$ depth 1 nodes is labeled with a different challenge y and has $7m+3$ children labeled x.
- The children are subtrees of size $(2m_\mu - 1) \dots (2m_2 - 1)$
- Each level has nodes labeled with the challenges x_i used in the i-th move of the recursive argument, and of degree $2m_{\mu-i+1} - 1$.

Given the above tree of transcripts, we are able to do a two-stage extraction: First, we invoke the witness-extended emulation of the recursive inner product argument. At this point, we either have a a non-trivial discrete logarithm relation, in which case we are done, or we have an accepting $\vec{r}$ for each y, x pair. In this case, we proceed with the second stage and repeat the extraction procedure for $\mu = 0$ to obtain either a witness for the original statement or a breach of the binding property of the commitment scheme.

We now point out that the size of the tree will be $O(N \cdot (2m)^\mu) \approx O(N^{2+\log_m 2})$ which is polynomial in the security parameter λ and invoke the Forking Lemma to complete the proof. □

5.6.7 Efficiency.

Square Root Communication. When we set $\mu = 0$, the argument above has a communication cost of $m'_1 + m'_2 + 2 + 1 + 3m$ commitments and $n + n' + 2$ field elements. Setting $m \approx \sqrt{\frac{N}{3}}$, $n \approx \sqrt{3N}$, $n' \approx \sqrt{7m}$, $m'_1 \approx 3\sqrt{\frac{m}{7}}$ and $m'_2 \approx 4\sqrt{\frac{m}{7}}$ we get a total communication complexity where the total number of group and field elements sent is as low as possible and approximately $2\sqrt{N}$ each. The main computational cost for the prover is computing the initial commitments, corresponding to $\frac{3mn}{\log n}$ group exponentiations. The prover can compute $t(X)$ using FFT-based techniques. Assuming that p is of a suitable form [Can89], the dominant number of multiplications for this process is $\frac{3}{2}mn\log m$. The main cost in the verification is computing $\vec{s}(X)$ given the description of the circuit which requires in the worst case Qn multiplications in $\mathbb{Z}_p$, considering arbitrary fan-in addition gates. In case of $O(N)$-size circuits with fan-in 2 gates, computing $\vec{s}(X)$ requires $O(N)$ multiplications. Evaluating $\vec{s}(x)$ requires $3N$ multiplications. The last verification equation costs roughly $\frac{(n+3m)}{\log n+3m}$ group exponentiations to the verifier.

$(\mu+1)$**-Root Communication.** We can reduce communication by using $\mu = O(1)$ iterations of the inner product argument. Choosing $m = N^{\frac{1}{\mu+1}}$, $n = N^{\frac{\mu}{\mu+1}}$ and $m_i = (\frac{N}{m})^{\frac{1}{\mu}}$ will give us a communication complexity of $4\mu N^{\frac{1}{\mu+1}}$ group elements and $2\mu N^{\frac{1}{\mu+1}}$ field elements. The prover's complexity is dominated by $\frac{6\mu N}{\log N}$ group exponentiations and fewer than $\frac{3N}{2\mu}\log N$ field multiplications. The verifier's cost is dominated by $\frac{2\mu N}{\log N}$ group exponentiations and $O(N)$ field multiplications.

Logarithmic Communication. By increasing the number of iteration of the inner product argument we can further reduce the communication complexity.

To minimise the communication, we set $\mu = \log N - 1$, $n = \frac{N}{2}$, $m = m_i = 2$, $m'_1 = 2$, $m'_2 = 3$ and $n' = 4$ in the above argument gives us $2\log N + 1$ moves. The total communication amounts to $4\log N + 7$ group elements and $2\log N + 6$ field elements. The prover computational cost is dominated by $12N$ group exponentiations, and $O(N)$ multiplications in $\mathbb{Z}_p$.The main verification cost is bounded by $4N$ group exponentiations and $O(N)$ multiplications in $\mathbb{Z}_p$.

Alternatively, we can optimise the computation while maintaining logarithmic communication by setting $\mu = \log N - \log\log 2N$, $m = \log N$, $n = \frac{N}{\log N}$, $n' \approx \sqrt{7\log N}$, $m'_1 \approx 3\sqrt{\frac{\log N}{7}}$, $m'_2 \approx 4\sqrt{\frac{\log N}{7}}$, $m_i = 2$ for $1 \leq i \leq \mu$. In this way we obtain a $2\log N - 2\log\log N + 1$ moves argument. With respect to the previous settings, we now save $2\log\log N$ moves by starting the inner product argument with a smaller statement. The resulting communication is at most $7\log N + \sqrt{7\log N}$ group elements and at most $2\log N + \sqrt{7\log N}$ field elements. Thus, the prover computation is dominated by $\frac{3N}{\log N}$ group exponentiations and $11N\log\log N$ field multiplications. For the verifier, it is bounded from above by $\frac{4N}{\log N \log\log N}$ group exponentiations and $O(N)$ field multiplications.

5.7 Implementation using Python

To verify the practicality of our construction we produced a proof of concept implementation in Python using the NumPy [Oli06] package. The more costly operations are executed natively: we use Petlib [Dan15] to outsource elliptic curve operations to the OpenSSL library, and also use a small C++ program to calculate the polynomial multiplication producing $\vec{t}(X)$ using NTL [Sho01]. Our implementation is single-threaded, but the operations performed are easily parallelisable. The implementation is open source and consists of approximately 3000 lines of Python and 100 lines of C++.

Our implementation accepts the circuit description format used by Pinocchio [PHGR13], which it preprocesses to remove addition and multiplication by constant gates, encoding them as a constraint table as in Section 5.3.1. Pinocchio also supports split gates, taking as input a single arithmetic wire and producing a fixed number of binary wires as outputs, so the binary wires correspond to the binary representation of the arithmetic wire. We handle split gates by adding appropriate multiplication gates and constraints to ensure binary wires can only carry zeroes or ones, and that their values scaled by the appropriate powers of 2 sum up to the gate's input.

5.7.1 Performance Comparison

We compared the performance of our implementation to that of Pinocchio [PHGR13] using the same set of circuits they used for their performance testing. The circuits implement multiplication of a vector by a fixed matrix, multiplication of two matrices, evaluation of a multivariate polynomial, and other applications for which we refer to [PHGR13]. We used an i5-4690K running Pinocchio under Windows 10 and our software under Ubuntu 14.04 for the tests.

We note here that Pinocchio operates in a pairing-based setting, using knowledge of exponent assumptions, whereas we operate in the discrete log setting. Even so, we feel the comparison is meaningful, as we are not aware of previous implementations of circuit-evaluation arguments in our setting.

From the comparison in Table 5.2, it is clear that our implementation is extremely competitive in terms of prover computation. For certain applications, the square root version outperformes Pinocchio by a factor larger than 10 in terms of prover speed. There is a significant amount of variance in terms of the speedups achieved. The worst cases are those where the number of constraints is high in comparison with the number of multi-

		This work										Pinocchio				
		Square Root					Logarithmic					(Constant)				
Application	Mult.	Key		Proof			Key		Proof			Key		Proof		
	Gates	Gen	Size	Prove	Verify	Size	Gen	Size	Prove	Verify	Size	Gen	Size	Prove	Verify	Size
		s	B	s	s	B	s	B	s	s	B	s	B	s	s	B
Vector Matrix	600	0.07	1120	0.38	0.25	6K	0.03	3872	0.55	0.31	3552	0.42	0.3M	0.23	.023	288
Product	1000	0.10	1440	0.76	0.61	8K	0.06	6464	1.05	0.67	3744	0.93	0.5M	0.53	.035	288
Matrix	347K	1.1	19K	14.7	3.4	76K	5.3	618K	49.9	22.9	5792	47.3	97.9M	167.4	.201	288
Product	1343K	2.7	37K	60.8	12.7	160K	18.6	2.2M	187.0	81.7	6496	170.4	374.8M	706.8	.503	288
Polynomial	203K	1.0	14K	30.0	2.1	88K	3.3	383K	53.1	14.0	5440	24.4	55.9M	146.8	.007	288
Evaluation	571K	1.7	24K	97.0	5.6	160K	8.3	962K	164.5	36.0	6272	60.2	156.8M	422.1	.007	288
Image	86K	0.7	9K	2.6	1.0	44K	1.5	171K	11.4	6.2	5120	15.2	23.6M	25.1	.007	288
Matching	278K	1.2	17K	7.4	2.9	72K	4.2	490K	34.3	18.1	5920	38.9	75.8M	88.8	.007	288
Shortest	366K	1.5	19K	9.3	3.7	52K	5.6	644K	45.6	23.9	5792	50.4	99.6M	130.7	.015	288
Paths	1400K	2.6	38K	35.1	12.6	72K	19.2	2.2M	169.8	84.0	6496	177.6	381.4M	523.3	.026	288
Gas	144K	0.8	12K	8.8	6.1	64K	2.3	271K	23.7	13.9	5440	22.6	39.6M	47.6	.007	288
Simulation	283K	1.2	17K	26.7	20.7	160K	4.3	503K	54.8	34.5	5920	45.9	77.7M	103.1	.007	288
SHA-1	24K	0.18	5K	3.7	3.3	24K	0.5	54K	6.5	4.3	4992	7.9	6.5M	9.0	.007	288

Table 5.2: Performance comparison between our implementation and Pinocchio. Pinocchio was set to use public verifiability and zero-knowledge.

plication gates: the calculation of $\vec{s}(X)$ is performed entirely in Python and thus becomes the dominant term in the computation. We expect that in a fully compiled implementation, optimisation would prevent this issue.

The logarithmic communication version is slower in comparison but still outperforms Pinocchio for most applications. The performance also becomes more even, as the constraints are irrelevant in the recursive part.

Our verification times are much higher than Pinocchio's, which can often verify circuit evaluation faster than native execution of an equivalent program. As with the prover, some speedups can be gained by moving to a compiled language, but we would still not expect to match Pinocchio's performance; our verification cost would still be linear. Our proofs are considerably larger as well, especially for the square root version.

Our key generation is simply a commitment key generation, and is not application-specific. Therefore, it can be easily amortised even across different circuits. For a circuit with N multiplication gates, the size of our commitment key is $\sqrt{N}$ elements for the square root version and $\frac{N}{\log N}$ for the log version. In comparison, Pinocchio's key generation is bound to specific circuits and produces keys of size $8N$. Thus, if the keys need to be communicated, our arguments are competitive in terms of total communication if the number of circuit evaluations is up to $\sqrt{N}$ for the square root version, and up to $\frac{N}{\log N}$ for the log version.

Chapter 6

Accountable Ring Signatures

Multiuser signatures (i.e. ring signatures and group signatures) are prominent cryptographic primitives offering a combination of privacy and authentication. They enable individual users to anonymously sign messages on behalf of a group of users. In ring signatures, the group, i.e. the ring, is chosen in an ad hoc manner by the signer. In group signatures, group membership is controlled by a group manager. Group signatures additionally enforce accountability by providing the group manager with a secret tracing key that can be used to identify the otherwise anonymous signer when needed.

Accountable ring signatures, introduced by Xu and Yung (CARDIS 2004), bridge the gap between the two notions. They provide maximal flexibility in choosing the ring, and at the same time maintain accountability by supporting a designated opener that can identify signers when needed.

In this chapter, we revisit accountable ring signatures and offer a formal security model for the primitive. Our model offers strong security definitions incorporating protection against maliciously chosen keys and at the same time flexibility both in the choice of the ring and the opener. We will give a generic construction using standard tools.

We will also give a highly efficient instantiation of our generic construction in the random oracle model by meticulously combining Camenisch's group signature scheme (CRYPTO 1997) with a generalization of the one-out-of-many proofs of knowledge by Groth and Kohlweiss (EUROCRYPT 2015). Our instantiation yields signatures of logarithmic size (in the size of the ring) while relying solely on the well-studied decisional Diffie-Hellman assumption. In the process, we offer a number of optimizations for the recent Groth and Kohlweiss one-out-of-many proofs, which may be useful for other applications.

Accountable ring signatures imply traditional ring and group signatures. We can therefore also obtain highly efficient instantiations of those primitives with signatures shorter than all existing ring signatures as well as existing group signatures relying on standard assumptions.

The research presented in this chapter was joint work with Jonathan Bootle, Andrea Cerulli, Essam Ghadafi, Jens Groth and Christophe Petit, and was published in [BCC$^+$15]. The author contributed to the investigation in the construction and proofs of the tree/list protocol (together with Andrea Cerulli), in the construction and proofs of the other NIZK protocols (together with Andrea Cerulli and Christophe Petit), the security proofs of the main Scheme (the move from CCA encryption to CPA coupled with straighline extractability is due to Jens Groth) and the discussion on revocation.

6.1 Introduction

Significant effort has been devoted to the study of signature schemes with privacy properties that allow a signer to remain anonymous within a set of users. Two prominent examples of anonymous signature schemes are ring signatures [RST01] and group signatures [CvH91]. Ring signatures allow a signer to choose any *ad hoc* set of users, i.e. a ring, and sign anonymously on behalf the ring. Group signatures also allow a signer to sign anonymously on behalf of a group of users but here group membership is controlled by a designated group manager. The advantage of group signatures is accountability; in case of abuse, the group manager can revoke anonymity and identify the signer.

Accountable ring signatures [XY04] bridge the gap between ring signatures and group signatures. They offer the flexibility of freely choosing the ring of users when creating a signature and at the same time enforce accountability by including an opener who can open a signature and reveal who signed it. The combination of flexibility and accountability allows applications where ring signatures or group signatures are less suitable. Consider, for instance, an online forum that wants to offer anonymity to users but also wants to be able to trace people who violate the code of conduct. A forum can achieve this by allowing user posts with accountable ring signatures where the owner is the specified opener. This system is decentralised and flexible since different fora can have their own opener keys and users do not have to register with each individual forum they post to. Another potential application is an auction system where bids are public but unsuccessful bidders want anonymity. Bidders sign bids with the seller as opener and at the end of the auctions the seller can disclose the winner in a verifiable way.

6.1.1 Contributions

We introduce a new security model for *accountable* ring signatures. The signer specifies, in addition to a set of users that could have produced the signature, the public key of an opening entity, which will be able to remove anonymity. This opening mechanism offers protection against misbehaving signers while at the same time not relying on a single, centralised group manager. Our security definitions are stringent and, when possible, incorporate protection against maliciously chosen keys.

We provide a generic construction of accountable ring signatures from standard cryptographic tools. We also give a concrete instantiation, combining ideas from Camenisch's group signature [Cam97] with a generalization of the one-out-of-many proof of knowledge of Groth and Kohlweiss [GK14]. The most efficient ring and group signatures [ACJT00, CL02, CKS09, BBS04, DKNS04, CG05, Ngu05, GK14] in the literature are in the random oracle model [FS87] and so is ours. However, the only other assumption we make is the hardness of the well-established decisional Diffie-Hellman problem.[1]

From a technical viewpoint, we offer two optimisations of Groth-Kohlweiss one-out-of-many proofs. One perspective on their proof system is that they form a binary tree and prove that one of the leaves is selected. We generalise their approach to n-ary trees, allowing us to fine-tune the parameters for better performance. For $N = n^m$, our optimisations reduce the number of group elements in the 1-out-of-N proof from $4m$ to $2m$ with little impact on the number of field elements or computational cost. Also, while their proofs can be used for ElGamal encryption, which is what we need for our scheme, this imposes an overhead in all parts of their protocol. We deploy more efficient Pedersen commitments in some parts of the proof, thus limiting the overhead of ElGamal.

The end result is an accountable ring signature scheme with efficient computation and very small signatures. Namely, for a ring with $N = \mathrm{poly}(\lambda)$ users, we obtain signatures of size approximately $\frac{5}{2}\lambda \log_2 N$ bits, which is smaller than all existing group and ring signatures based on standard assumptions.

6.2 Defining Accountable Ring Signatures

An accountable ring signature (ARS) scheme over a PPT setup Setup is a tuple of polynomial-time algorithms $(\mathrm{OKGen}, \mathrm{UKGen}, \mathrm{Sign}, \mathrm{Vfy}, \mathrm{Open}, \mathrm{Judge})$.

$\mathrm{Setup}(1^\lambda)$: Given the security parameter, produces public parameters pp used (sometimes implicitly) by the rest of the scheme. The public parameters define key spaces $\mathrm{PK}, \mathrm{DK}, \mathrm{VK}, \mathrm{SK}$ with efficient algorithms for sampling and deciding membership.

$\mathrm{OKGen}(pp)$: Given the public parameters pp, produces a public key $pk \in \mathrm{PK}$ and secret key $dk \in \mathrm{DK}$ for an opener. Without loss of generality, we assume dk defines pk deterministically and write $pk = \mathrm{OKGen}(pp, dk)$ when computing pk from dk.

$\mathrm{UKGen}(pp)$: Given the public parameters pp, produces a verification key $vk \in \mathrm{VK}$ and a secret signing key $sk \in \mathrm{SK}$ for a user. We can assume sk deterministically determines vk and write $vk = \mathrm{UKGen}(pp, sk)$ when computing vk from sk.

[1] An important advantage of working over a discrete logarithm group is that so do many standard signature schemes, e.g., DSS. We therefore already have many users with suitable public verification keys in a standard cyclic group, e.g., NIST's 256-bit elliptic curve group P-256.

$\mathsf{Sign}(pk, m, R, sk)$**:** Given an opener's public key, a message, a ring (i.e. a set of verification keys) and a secret key, produces a ring signature σ. The algorithm returns the error symbol $\perp$ if the inputs are malformed, i.e., if $pk \notin \mathrm{PK}, R \not\subset \mathrm{VK}, sk \notin \mathrm{SK}$ or $vk = \mathrm{UKGen}(pp, sk) \notin R$.

$\mathsf{Vfy}(pk, m, R, \sigma)$**:** Given an opener's public key, a message, a ring and a signature, returns 1 if accepting the signature and 0 otherwise. We assume the algorithm always returns 0 if the inputs are malformed, i.e., if $pk \notin \mathrm{PK}$ or $R \not\subset \mathrm{VK}$.

$\mathsf{Open}(m, R, \sigma, dk)$**:** Given a message, a ring, a ring signature and an opener's secret key, returns a verification key vk and a proof ψ that the owner of vk produced the signature. If any of the inputs is invalid, i.e., $dk \notin \mathrm{DK}$ or σ is not a valid signature using $pk = \mathrm{OKGen}(pp, dk)$, the algorithm returns $\perp$.

$\mathsf{Judge}(pk, m, R, \sigma, vk, \psi)$**:** Given an opener's public key, a message, a ring, a signature, a verification key and a proof, returns 1 if accepting the proof and 0 otherwise. We assume the algorithm returns 0 if σ is invalid or $vk \notin R$.

An accountable ring signature scheme should be correct, fully unforgeable, anonymous and traceable as defined below.

Definition 25 (Perfect correctness)**.** *An accountable ring signature scheme is perfectly correct if for any PPT adversary $\mathcal{A}$*

$$\Pr\left[\begin{array}{c} pp \leftarrow \mathrm{Setup}(1^\lambda); (vk, sk) \leftarrow \mathrm{UKGen}(pp); \\ (pk, m, R) \leftarrow \mathcal{A}(pp, sk); \sigma \leftarrow \mathrm{Sign}(pk, m, R, sk) \ : \\ \text{If } pk \in \mathrm{PK}, R \subset \mathrm{VK}, vk \in R \text{ then } \mathrm{Vfy}(pk, m, R, \sigma) = 1 \end{array}\right] = 1.$$

We remark that correctness of the opening algorithm (w.r.t. an honestly generated opener key) is implied by the other requirements.

Full unforgeability ensures that an adversary, who may control the opener, can neither falsely accuse an honest user of producing a ring signature nor forge ring signatures on behalf of an honest ring. The former should hold even when all other users in the ring are corrupt. This requirement combines the non-frameability of group signatures [BSZ05] and the unforgeability of ring signatures [BKM09] requirements.

Definition 26 (Full Unforgeability)**.** *An accountable ring signature scheme is fully unforgeable if for any PPT adversary $\mathcal{A}$*

$$\Pr\left[\begin{array}{c} pp \leftarrow \mathrm{Setup}(1^\lambda); (pk, vk, m, R, \sigma, \psi) \leftarrow \mathcal{A}^{\mathrm{UKGen},\mathrm{Sign},\mathrm{Reveal}}(pp) : \\ \Big(vk \in Q_{\mathrm{UKGen}} \setminus Q_{\mathrm{Reveal}} \wedge (pk, vk, m, R, \sigma) \notin Q_{\mathrm{Sign}} \\ \wedge \mathrm{Judge}(pk, m, R, \sigma, vk, \psi) = 1\Big) \\ \vee \Big(R \subset Q_{\mathrm{UKGen}} \setminus Q_{\mathrm{Reveal}} \wedge (pk, \cdot, m, R, \sigma) \notin Q_{\mathrm{Sign}} \\ \wedge \mathrm{Vfy}(pk, m, R, \sigma) = 1\Big) \end{array}\right] \approx 0.$$

- UKGen *runs* $(vk, sk) \leftarrow \mathrm{UKGen}(pp)$ *and returns vk.* Q_{UKGen} *is the set of verification keys vk that have been generated by this oracle.*
- Sign *is an oracle that on query* (pk, vk, m, R) *returns* $\sigma \leftarrow \mathrm{Sign}(pk, m, R, sk)$ *if* $vk \in R \cap Q_{\mathrm{UKGen}}$. Q_{Sign} *contains the queries and responses* (pk, vk, m, R, σ).
- Reveal *is an oracle that when queried on* $vk \in Q_{\mathrm{UKGen}}$ *returns the corresponding signing key sk.* Q_{Reveal} *is the list of verification keys vk for which the corresponding signing key has been revealed.*

Anonymity ensures that a signature does not reveal the identity of the ring member who produced it without the opener explicitly wanting to open the particular signature. We allow the adversary to choose the secret signing keys of the users which implies anonymity against full key exposure attacks [BKM09] where the users' secret signing keys have been revealed. Our definition also captures unlinkability as used in [XY04]: if an adversary can link signatures by the same signer, it can break anonymity.

Definition 27 (Anonymity)**.** *An accountable ring signature scheme is anonymous if for any PPT adversary $\mathcal{A}$*

$$\Pr\left[\begin{array}{c} pp \leftarrow \mathrm{Setup}(1^\lambda); b \leftarrow \{0,1\}; (pk, dk) \leftarrow \mathrm{OKGen}(pp) : \\ \mathcal{A}^{\mathrm{Chal}_b,\mathrm{Open}}(pp, pk) = b \end{array}\right] \approx \frac{1}{2}.$$

- Chal_b *is an oracle that the adversary can only call once. On query* (m, R, sk_0, sk_1) *it runs* $\sigma_0 \leftarrow \mathrm{Sign}(pk, m, R, sk_0)$; $\sigma_1 \leftarrow \mathrm{Sign}(pk, m, R, sk_1)$. *If* $\sigma_0 \neq \perp$ *and* $\sigma_1 \neq \perp$ *it returns* σ_b, *otherwise it returns* $\perp$.
- Open *is an oracle that on a query* (m, R, σ) *returns* $\mathrm{Open}(m, R, \sigma, dk)$. *If* σ *was*

obtained by calling Chal_b *on* (m,R)*, the oracle returns* $\perp$*.*

Traceability ensures that the specified opener can always identify the ring member who produced a signature and that she is able to produce a valid proof for her decision. We allow the adversary full control in specifying the decryption key of the opener, as well as the ring R.

Definition 28 (Traceability)**.** *An accountable ring signature scheme is traceable if for any PPT adversary* $\mathcal{A}$

$$\Pr\left[\begin{array}{c} pp \leftarrow \mathrm{Setup}(1^\lambda); (dk,m,R,\sigma) \leftarrow \mathcal{A}(pp); \\ pk \leftarrow \mathrm{OKGen}(pp,dk); (vk,\psi) \leftarrow \mathrm{Open}(m,R,\sigma,dk): \\ \mathrm{Vfy}(pk,m,R,\sigma) = 1 \ \wedge \ \mathrm{Judge}(pk,m,R,\sigma,vk,\psi) = 0 \end{array}\right] \approx 0.$$

Tracing soundness ensures that a signature cannot trace to two different users; only one person can be identified as the signer even when all users as well as the opener are fully corrupt. Similarly to the setting of group signatures [SSE+12], this requirement is vital for some applications, e.g., where users might be rewarded for signatures they produced, or to avoid shifting blame when signatures are used as evidence of abuse. We allow the adversary full control in specifying the keys of the opener, the ring R and all the users. The setting of tracing soundness is different than that of the traceability game, but is compatible with that of full unforgettability. However, in the interest of readability, we cover it separately.

Definition 29 (Tracing Soundness)**.** *An accountable ring signature scheme satisfies tracing soundness if for any PPT adversary* $\mathcal{A}$

$$\Pr\left[\begin{array}{c} pp \leftarrow \mathrm{Setup}(1^\lambda); (m,\sigma,R,pk,vk_1,vk_2,\psi_1,\psi_2) \leftarrow \mathcal{A}(pp): \\ \forall i \in \{1,2\},\ \mathrm{Judge}(pk,m,R,\sigma,vk_i,\psi_i) = 1 \wedge \ vk_1 \neq vk_2 \end{array}\right] \approx 0.$$

6.2.1 Ring and Group Signatures from Accountable Ring Signatures

We will now relate accountable ring signatures to ring signatures and group signatures by showing that the latter are implied by accountable ring signatures.

RING SIGNATURES. Traditional ring signatures [RST01] do not have an opener and their security requires anonymity of the signer and unforgeability [RST01, BKM09]. By modifying the signing algorithm to produce a freshly sampled opener key and attach it to the signature, we obtain a traditional ring signature scheme from an accountable ring signature.

As the public key is freshly sampled, the opening and judge algorithms are ignored no user can derive the private key required to open signatures and produce proofs. Correctness and anonymity follow from those of the accountable ring signature, whereas unforgeability is implied by full unforgeability and traceability.

GROUP SIGNATURES. Bellare et al. [BMW03] defined group signatures for static groups, where the population of the group is fixed once and for all at the setup time, and where the group manager additionally acts as the designated opener. Besides correctness, their model requires full anonymity and full traceability. The latter requires that an adversary in possession of the group master secret key who can corrupt members of the group, cannot produce a new signature that does not trace to a user in set of corrupt users. An accountable ring signature satisfying our security definitions gives rise to a group signature scheme as follows: We fix the group manager as the designated opener and set the corresponding decryption key as the group master secret key *gmsk* used as the tracing key. In the setup, the group members generate their personal key pairs and the group manager publishes the ring containing the public keys of the members as part of the group signature public key. Group membership is enforced by the manager's control over the ring. Group signatures are then just accountable ring signatures w.r.t. this ring. Full anonymity follows from the anonymity of the accountable ring signature scheme, whereas full traceability follows from full unforgeability and traceability. In Sect. 7.4.3, we give a detailed example based on the generic accountable ring signature construction of the following section.

The group public key in our scheme is quite large since it grows linearly in the number of members. However, the cost of transmitting the public key that can be amortised over many signatures: a user needs to obtain the public key only once in the static version, and once per epoch in the dynamic one (furthermore, the dynamic case is also well suited to delta-based solutions). An advantage of a group signature scheme based on ARS on the other hand is that it can easily be made dynamic, something that we will cover in detail in the next chapter. The group manager can enrol or remove users by adding or deleting their verification keys from the group public key [DKNS04]. In the dynamic group signature scheme, the group public key is changing and group signatures must be verified against the group as it was at the time of signing, but for scenarios where the group is not changing too often or where great flexibility is desired this is a price worth paying.

6.3 Constructing Accountable Ring Signatures

6.3.1 Building Blocks

Our generic construction (shown in Fig. 6.1) uses a one-way function f (Sect. 3.2.2), an IND-CPA secure public-key encryption scheme $(\mathrm{PKEGen}, \mathcal{E}, \mathcal{D})$ (Sect. 3.2.2), a signature of knowledge (Sect. 3.4.7), and a non-interactive zero-knowledge proof (Sect. 3.4.5), all of which share the same setup gk. The setup gk defines domain SK and range VK for f, and key, message, randomness and ciphertext spaces $\mathrm{PK}, \mathrm{DK}, \mathrm{M}, \mathrm{Rnd}, \mathrm{C}$ for the encryption scheme. The range of f and the message space of the encryption scheme need to be compatible such that $\mathrm{VK} \subseteq \mathrm{M}$. We describe the relations w.r.t the signature of knowledge and the zero knowledge proof operate in the next section.

6.3.2 Design

The idea is that an opener will have a key pair for the encryption scheme and the user will have a secret key sk and corresponding verification key $vk = f(sk)$. To sign a message m w.r.t. a ring R, the signer first encrypts her verification key under the opener's public key and provides a signature of knowledge on m proving the ciphertext encrypts a verification key in the ring and that she knows the secret key associated with the encrypted verification key. To open a signature, the opener decrypts the ciphertext to obtain the user's verification key and provides an NIZK proof of correct decryption.

The relations $\mathcal{R}_{\mathrm{sig}}$ and $\mathcal{R}_{\mathrm{open}}$ associated with the signature of knowledge and the NIZK system, respectively, are as follows:

$$\mathcal{R}_{\mathrm{sig}} := \left\{ \begin{array}{c} \big((pk, R, c), (sk, r)\big) : \\ R \subset \mathrm{VK} \ \land \ vk := f(sk) \in R \ \land \ c = \mathcal{E}(pk, vk; r) \end{array} \right\}.$$

$$\mathcal{R}_{\mathrm{open}} := \left\{ \begin{array}{c} \big((pk, c, vk), dk\big) : \\ pk = \mathrm{PKEGen}(gk; dk) \in \mathrm{PK} \ \land \ vk = \mathcal{D}(dk, c) \ \land \ vk \in \mathrm{VK} \end{array} \right\}.$$

6.3.3 Security

Lemma 2. *The accountable ring signature scheme in Fig. 6.1 is anonymous if* SoK *for* $\mathcal{R}_{\mathrm{sig}}$ *is SimExt secure, the NIZK for* $\mathcal{R}_{\mathrm{open}}$ *is zero knowledge, and* $(\mathrm{PKEGen}, \mathcal{E}, \mathcal{D})$ *is IND-CPA secure.*

Proof. We start by replacing the algorithm SoKSetup of the signature of knowledge with SoKSimSetup, and when answering the challenge query, we use SoKSimSign instead of SoKSign. By the SimExt security of the SoK, the adversary has a negligible probability in

```
$\mathrm{Setup}(1^\lambda)$
    $gk \leftarrow \mathcal{G}(1^\lambda); \mathsf{crs} \leftarrow \mathrm{CRSGen}(gk)$
    $\mathsf{crs}_{\mathrm{SoK}} \leftarrow \mathrm{SoKSetup}(gk)$
    Return $pp := (gk, \mathsf{crs}_{\mathrm{SoK}}, \mathsf{crs})$

$\mathrm{OKGen}(pp)$
    $(pk, dk) \leftarrow \mathrm{PKEGen}(gk)$
    Return $(pk, dk)$

$\mathrm{Open}(m, R, \sigma, dk)$
    $pk \leftarrow \mathrm{OKGen}(pp; dk)$
    If $\mathrm{Vfy}(pk, m, R, \sigma) = 0$ return $\perp$
    Parse $\sigma$ as $(c, \sigma_{\mathrm{SoK}})$
    $vk := \mathcal{D}(dk, c)$
    $\psi \leftarrow \mathsf{Prove}(\mathsf{crs}, (pk, c, vk), dk)$
    Return $(vk, \psi)$

$\mathrm{UKGen}(pp)$
    $sk \leftarrow \mathrm{SK}$; $vk := f(sk)$; Return $(vk, sk)$

$\mathrm{Sign}(pk, m, R, sk)$
    $vk \leftarrow f(sk); r \leftarrow \mathrm{Rnd}; c \leftarrow \mathcal{E}(pk, vk; r)$
    $\sigma_{\mathrm{SoK}} \leftarrow \mathrm{SoKSign}(\mathsf{crs}_{\mathrm{SoK}}, (pk, R, c), (sk, r), m)$
    Return $\sigma := (c, \sigma_{\mathrm{SoK}})$

$\mathrm{Vfy}(pk, m, R, \sigma)$
    Parse $\sigma$ as $(c, \sigma_{\mathrm{SoK}})$
    Return $\mathsf{SoKVerify}(\mathsf{crs}_{\mathrm{SoK}}, (pk, R, c), m, \sigma_{\mathrm{SoK}})$

$\mathrm{Judge}(pk, m, R, \sigma, vk, \psi)$
    If $\mathrm{Vfy}(pk, m, R, \sigma) = 0$ return 0
    Parse $\sigma$ as $(c, \sigma_{\mathrm{SoK}})$
    Return $\mathrm{PVfy}(\mathsf{crs}, (pk, c, vk), \psi)$
```

Figure 6.1: A generic construction for Accountable Ring Signatures

distinguishing between the two settings. This ensures that the signature of knowledge σ_{SoK} reveals no information about the underlying witness.

Next, we replace the algorithm CRSGen of the NIZK system with SimCRSGen and when answering opening queries, we use SimProve instead of Prove. By the zero-knowledge property of the NIZK system, the adversary has a negligible probability in distinguishing between the two settings.

Now, we modify the Open oracle into Open$'$ such that instead of decrypting the ciphertext, we run $\mathsf{SoKExtract}$ to extract the verification key vk from the signature of knowledge σ_{SoK}. By the SimExt security of the signature of knowledge, with overwhelming probability in each query, we get the same vk as the plaintext of c.

As we are no longer using the decryption algorithm, by the IND-CPA security of the encryption scheme, the probability of $\mathcal{A}$ winning the anonymity game is close to $\frac{1}{2}$. □

Lemma 3. *The accountable ring signature scheme in Fig. 6.1 is fully unforgeable if* SoK *for $\mathcal{R}_{\mathrm{sig}}$ is SimExt secure, the NIZK for $\mathcal{R}_{\mathrm{open}}$ is sound, and f is one-way.*

Proof. We start by running the Setup algorithm as normal with the exception that here we replace SoKSetup with $\mathsf{SoKSimSetup}$. We forward crs to the adversary. By the simulatability of the signature of knowledge, the adversary has a negligible probability in distinguishing between the two settings. From now on, we use SoKSimSign instead of SoKSign when answering Sign queries. The adversary can win in two ways:

- **Case I:** The adversary forges a valid ring signature on a message m w.r.t. an honest ring R where $(pk, \cdot, m, R, \sigma) \notin Q_{\mathrm{Sign}}$. By the SimExt security of the signature of knowledge, we can extract a valid witness for the statement $(pk, R, c) \in \mathcal{L}_{\mathcal{R}_{\text{sig}}}$ from which we obtain (vk, sk) such that $vk := f(sk) \in R$. We use this to break the one-wayness of the function f which contradicts the security of the function f.

- **Case II:** The adversary outputs a valid ring signature $\sigma := (c, \sigma_{\text{SoK}})$ on a message m w.r.t. a ring R and a proof ψ that the honest user with key vk produced the signature while such user never did so.

  We start by guessing the user the adversary is going to frame. We have a probability $\frac{1}{\eta(\lambda)}$ of guessing correctly, where $\eta(\lambda)$ is a polynomial representing an upper bound on the number of honest users $\mathcal{A}$ uses in the game. By the soundness of the NIZK system, ψ is a proof for a valid statement $(pk, c, vk) \in \mathcal{L}_{\mathcal{R}_{\text{open}}}$. In the game, we abort if $\mathcal{A}$ asks for the secret key of the user we guessed. For all other honest users, we have chosen their key pairs ourselves and thus know their secret keys.

  Again, by the simulation extractability of the signature of knowledge, with overwhelming probability, we can extract a valid witness for the statement $(pk, R, c) \in \mathcal{L}_{\mathcal{R}_{\text{sig}}}$ from σ_{SoK}, from which we obtain (vk, sk) such that $vk := f(sk) \in R$. We use this to break the one-wayness of f which contradicts the security of the function f.

□

Lemma 4. *The accountable ring signature scheme in Fig. 6.1 is traceable, if* SoK *for* $\mathcal{R}_{\text{sig}}$ *is SimExt secure, the NIZK for* $\mathcal{R}_{\text{open}}$ *is complete, and is* $(\text{PKEGen}, \mathcal{E}, \mathcal{D})$ *perfectly correct.*

.

Proof. By the security of the signature of knowledge, we are able to extract a valid witness from σ_{SoK} part of the valid signature $\sigma = (c, \sigma_{\text{SoK}})$ the adversary outputs. The witness thus satisfies $vk = f(sk)$ where $vk \in R \subset \text{VK}$, $pk \in \text{PK}$ and $c = \mathcal{E}(pk, vk; r)$ for some $r \in \text{Rnd}$ and $sk \in \text{SK}$.

Since $pk = \text{PKEGen}(gk; dk)$, we see from $pk \neq \perp$ that $dk \in \text{DK}$. Correctness of the encryption algorithm implies that $\mathcal{D}(dk, c) = vk$, which is the first part of the opening algorithm's output. Now the opening algorithm has a statement (pk, c, vk) and a corresponding witness dk. By the completeness of the NIZK proof system, ψ will verify correctly. This means that the Judge algorithm will output 1 which is a contradiction. □

Lemma 5. *The construction satisfies tracing soundness if* SoK *is SimExt secure, the NIZK proof system is sound and the encryption scheme is perfectly correct.*

Proof. The SimExt security of the signature of knowledge ensures that from any signature σ_{SoK} (w.r.t. a statment s) output by the adversary, we can extract a valid witness w such that $(s,w) \in \mathcal{R}_{\mathrm{sig}}$ which eliminates the case that the adversary forges a signature for a statement $s^* \notin \mathcal{L}_{\mathcal{R}_{\mathrm{sig}}}$. If this is not the case, we can use such an adversary to construct another adversary against the SimExt security of the signature of knowledge.

The soundness of the NIZK system for the relation $\mathcal{R}_{\mathrm{open}}$ ensures that ciphertext c contained in the ring signature decrypts to vk, which eliminates the case that the adversary can produce a proof ψ for a statement $s^* \notin \mathcal{L}_{\mathcal{R}_{\mathrm{open}}}$. Finally, the perfect correctness of the public-key encryption scheme (which is regarded as a perfectly-binding commitment scheme) ensures that a ciphertext has a unique decryption. □

Theorem 9. *The accountable ring signature construction in Fig. 6.1 is perfectly correct, anonymous, fully unforgeable, traceable, and satisfies tracing soundness if* SoK *for* $\mathcal{R}_{\mathrm{sig}}$ *is SimExt secure, the NIZK for* $\mathcal{R}_{\mathrm{open}}$ *is complete, sound and zero knowledge,* $(\mathrm{PKEGen}, \mathcal{E}, \mathcal{D})$ *is perfectly correct and IND-CPA secure, and f is one-way. .*

Proof. Perfect correctness follows from that of the building blocks and is easy to verify. Lemmata 2, 3, 4 and 5 complete the rest of the proof. □

Since all the building blocks can be constructed from (doubly enhanced [Gol11]) trapdoor permutations, we get as a corollary that trapdoor permutations imply the existence of accountable ring signatures.

6.4 Efficient Instantiation

We give here an efficient instantiation of the generic construction from Fig. 6.1. The instantiation is secure in the random oracle model under the well-established DDH assumption. We instantiate f with group exponentiation and the IND-CPA encryption scheme with El-Gamal. We will get the Signature of Knowledge and NIZK proof for the relations $\mathcal{R}_{\mathrm{sig}}$ and $\mathcal{R}_{\mathrm{open}}$ by applying the Fiat-Shamir transform to suitable Σ-protocols for these relations. Thanks to the straightline f-Extractability of our instantiation of the signature of knowledge, we can answer the adversary's Open queries in the anonymity game by extracting $vk = f(sk)$ from σ_{SoK} without rewinding.

For all Σ-protocols, the setup includes the group description gk and the common reference string $crs := (ck, ek)$, where $ck \leftarrow \mathrm{CGen}(gk)$, $(ek, \tau) \leftarrow \mathrm{PKEGen}(gk)$ and $ek = g^{\tau}$

for $\tau \leftarrow \mathbb{Z}_p^*$, for the Pedersen commitment scheme and the ElGamal encryption scheme, respectively.

6.4.1 Committed bits

We first give a Σ-protocol for a commitment B having an opening consisting of sequences of bits, where in each sequence there is exactly one 1. More precisely, we give in Fig. 6.2 a Σ-protocol $(\mathcal{G}_{\mathsf{crs}}, \mathcal{P}_1, \mathcal{V}_1)$ for the relation

$$\mathcal{R}_1 = \left\{ \begin{array}{c} (B, (b_{0,0}, \ldots, b_{m-1,n-1}, r)) : \\ (\forall i, j : b_{j,i} \in \{0,1\}) \wedge (\forall j : \sum_{i=0}^{n-1} b_{j,i} = 1) \wedge B = \mathrm{Com}_{ck}(b_{0,0}, \ldots, b_{m-1,n-1}; r) \end{array} \right\}$$

The main idea is to prove that $b_{j,i}(1 - b_{j,i}) = 0$ for all i, j, and also that $\sum_{i=1}^{n} b_{j,i} = 1$.

$$\begin{array}{lcl}
\underline{\mathcal{P}_1(gk, \mathsf{crs}, B, (b_{0,0}, \ldots, b_{m-1,n-1}, r))} & & \underline{\mathcal{V}_1(gk, \mathsf{crs}, B)} \\
r_A, r_C, r_D, a_{j,1}, \ldots, a_{j,n-1} \leftarrow \mathbb{Z}_p & & \\
\forall j : a_{j,0} := -\sum_{i=1}^{n-1} a_{j,i} & A, C, D & \\
A := \mathrm{Com}_{ck}(a_{0,0}, \ldots, a_{m-1,n-1}; r_A) & \xrightarrow{\hspace{6em}} & \\
C := \mathrm{Com}_{ck}(\{a_{j,i}(1-2b_{j,i})\}_{j,i=0}^{m-1,n-1}; r_C) & & \text{Accept if and only if} \\
D := \mathrm{Com}_{ck}(-a_{0,0}^2, \ldots, -a_{m-1,n-1}^2; r_D) & x \leftarrow \{0,1\}^{\lambda} & \quad A, B, C, D \in \mathbb{G} \\
 & \xleftarrow{\hspace{6em}} & \quad f_{0,1}, \ldots, f_{m-1,n-1}, z_A, z_C \in \mathbb{Z}_p \\
\forall j, i : f_{j,i} := b_{j,i}x + a_{j,i} & & \quad \forall j : f_{j,0} := x - \sum_{i=1}^{n-1} f_{j,i} \\
z_A := rx + r_A & f_{0,1}, f_{1,1}, \ldots, f_{m-1,n-1}, z_A, z_C & \quad B^x A = \mathrm{Com}_{ck}(f_{0,0}, \ldots, f_{m-1,n-1}; z_A) \\
z_C := r_C x + r_D & \xrightarrow{\hspace{6em}} & \quad C^x D = \mathrm{Com}_{ck}(\{f_{j,i}(x - f_{j,i})\}_{j,i=0}^{m-1,n-1}; z_C)
\end{array}$$

Figure 6.2: Σ-protocol for relation $\mathcal{R}_1$.

Lemma 6. *The* Σ-protocol *in Fig. 6.2 is perfectly complete, perfect SHVZK, computational 3-special sound and has quasi-unique responses, under the discrete logarithm assumption.*

Proof. Perfect completeness follows by inspection. The SHVZK simulator, given a challenge x, can simulate the transcript by picking $f_{0,1}, \ldots, f_{m-1,n-1}, z_A, z_C \leftarrow \mathbb{Z}_p$, $C \leftarrow \mathbb{G}$ and computing $f_{j,0} := x - \sum_{i=1}^{n-1} f_{j,i}$, $A := \mathrm{Com}_{ck}(f_{0,0}, \ldots, f_{m-1,n-1}, z_A)B^{-x}$, $D = \mathrm{Com}_{ck}(\{f_{i,j}(x - f_{i,j})\}_{i,j=0}^{m-1,n-1}; z_C)C^{-x}$. In both simulations and real proofs, $f_{0,1}, \ldots, f_{m-1,n-1}, z_A, z_C$ and C are independent, uniformly random and uniquely determine $\{f_{j,0}\}_{j=0}^{m-1}, A, D$, so the simulation is perfect. We also have quasi-unique responses, since two different valid answers $f_{0,1}, \ldots, f_{m-1,n-1}, z_A, z_C$ and $f'_{0,1}, \ldots, f'_{m-1,n-1}, z'_A, z'_C$ to one challenge would break the binding property of $B^x A$ and $C^x D$.

We prove 3-special soundness in three parts. First, we show that any answers to 3 (actually 2) different challenges provide an opening of B. Second, we show that these answers imply that committed values are bits. Finally, we show that they imply that the sum of the committed values is 1. For the first part, suppose that a prover has answered two different challenges x,x' correctly with answers $(f_{0,1},\dots,f_{m-1,n-1},z_A,z_C)$ and $(f'_{0,1},\dots,f'_{m-1,n-1},z'_A,z'_C)$. Since we have $B^xA = \mathrm{Com}_{ck}(f_{0,0},\dots,f_{m-1,n-1};z_A)$ and $B^{x'}A = \mathrm{Com}_{ck}(f'_{0,0},\dots,f'_{m-1,n-1};z'_A)$, from the first verification equation we have $B^{x-x'} = \mathrm{Com}_{ck}(f_{0,0}-f'_{0,0},\dots,f_{m-1,n-1}-f'_{m-1,n-1};z_A-z'_A)$. Thus $b_{i,j} = \frac{f_{i,j}-f'_{i,j}}{x-x'}$, with $r = \frac{z_A-z'_A}{x-x'}$, gives us an opening of B. The first verification equation also gives an opening $(a_0,\dots,a_0;r_A)$ of A using $a_{j,i} = f_{j,i} - xb_{j,i}$ and $r_A = z_A - xr$. Note that by the binding properties of the commitment scheme, the prover cannot know a second opening of A or B, and must respond to any challenge with $f_{j,i} = b_{j,i}x + a_{j,i}$. We can get openings of C and D to values $c_{j,i}$, $d_{j,i}$ from the second equation in a similar way.

By the second verification equation, the values satisfy $c_{j,i}x + d_{j,i} = f_{j,i}(x - f_{j,i}) = b_{j,i}(1-b_{j,i})x^2 + (1-2b_{j,i})a_{j,i}x - a^2_{j,i}$. If this holds for three different x,x' and x'' then the polynomials are identical. So, $b_{j,i}(1-b_{j,i}) = 0$ and $b_{j,i} \in \{0,1\}$ for all i,j.

By construction we have $\sum_{i=0}^{n-1} f_{j,i} = \sum_{i=0}^{n-1} b_{j,i}x + \sum_{i=0}^{n-1} a_{j,i} = x$ for all $j = 0,\dots,m-1$. This holds for two challenges x and x'. Therefore $\sum_{i=0}^{n-1} b_{j,i} = 1$. □

6.4.2 List Containing Encryption of 1

We now describe a Σ-protocol that a list of N ElGamal ciphertexts $(c_0,\dots,c_{N-1})$ includes an encryption of 1. More precisely, we give a Σ-protocol $(\mathcal{G}_{\mathsf{crs}},\mathcal{P}_2,\mathcal{V}_2)$ (see Fig. 6.3) for the relation:

$$\mathcal{R}_2 = \left\{ \ ((\{c_i\}_{i=0}^{N-1}),(\ell,r)) : (\forall i, c_i \in \mathbb{G}^2) \wedge \ell \in \{0,\dots,N-1\} \wedge c_\ell = \mathcal{E}_{ek}(1;r) \ \right\}$$

This generalises easily to other homomorphic encryption and commitment schemes.

Since we can pad the list with copies of the last ciphertext (at little extra cost in the protocol), we may assume $N = n^m$. We will later discuss the efficiency implications of different choices of n. The idea behind our Σ-protocol is to prove knowledge of an index ℓ for which the product $\prod_{i=0}^{N-1} c_i^{\delta_{\ell,i}}$ is an encryption of 1, where as usual $\delta_{\ell,i} = 1$ when $i = \ell$ and $\delta_{\ell,i} = 0$ otherwise. We have $\delta_{\ell,i} = \prod_{j=0}^{m-1} \delta_{\ell_j,i_j}$, where $\ell = \sum_{j=0}^{m-1} \ell_j n^j$ and $i = \sum_{j=0}^{m-1} i_j n^j$ are the n-ary representations of ℓ and i respectively.

The prover first commits to m sequences of n bits $(\delta_{\ell_j,0},\dots,\delta_{\ell_j,n-1})$. It runs the Σ-protocol in Fig. 6.2 to prove that the commitment is well-formed. On receiving a challenge x, the prover discloses elements $f_{j,i} = \delta_{\ell_j,i}x + a_{j,i}$ as in Fig. 6.2. Observe that for every $i \in \{0, \ldots, N-1\}$, the product $\prod_{j=0}^{m-1} f_{j,i_j}$ is the evaluation at x of the polynomial $p_i(x) = \prod_{j=0}^{m-1}(\delta_{\ell_j,i}x + a_{j,i})$. For $0 \leq i \leq N-1$, we have:

$$p_i(x) = \prod_{j=0}^{m-1} \delta_{\ell_j,i_j} x + \sum_{k=0}^{m-1} p_{i,k}x^k = \delta_{\ell,i}x^m + \sum_{k=0}^{m-1} p_{i,k}x^k, \tag{6.1}$$

for some coefficients $p_{i,k}$ depending on ℓ and $a_{j,i}$. Note that $p_{i,k}$ can be computed by the prover independently of x, and that $p_\ell(x)$ is the only degree m polynomial amongst $p_0(x), \ldots, p_{N-1}(x)$. From these coefficients and some random noise values ρ_k, the prover computes ciphertexts $G_k := \prod_{i=0}^{N-1} c_i^{p_{i,k}} \cdot \mathcal{E}_{ek}(1; \rho_k)$ and includes them in the initial message. These ciphertexts are then used to cancel out the low degree terms in (6.1). Namely, if c_ℓ is an encryption of 1, the following product is an encryption of 1 for any x

$$\prod_{i=0}^{N-1} c_i^{\prod_{j=0}^{m-1} f_{j,i_j}} \cdot \prod_{k=0}^{m-1} G_k^{-x^k} = \left(\prod_{i=0}^{N-1} c_i^{\delta_{\ell,i}}\right)^{x^m}.$$

$\underline{\mathcal{P}_2(gk, \mathsf{crs}, (c_0, \ldots, c_{N-1}), (\ell, r))}$		$\underline{\mathcal{V}_2(gk, \mathsf{crs}, (c_0, \ldots, c_{N-1}))}$
$r_B, \rho_k \leftarrow \mathbb{Z}_p$		
$B := \mathrm{Com}_{ck}(\delta_{\ell_{0,0}}, \ldots, \delta_{\ell_{m-1,n-1}}; r_B)$		
$(A, C, D) \leftarrow \mathcal{P}_1(gk, \mathsf{crs}, B, (\{\delta_{\ell_j,i}\}_{j,i=0}^{m-1,n-1}, r_B))$		
For $k = 0, \ldots, m-1$	$\xrightarrow{A, B, C, D, \{G_k\}_{k=0}^{m-1}}$	Accept if and only if
$\quad G_k = \prod_{i=0}^{N-1} c_i^{p_{i,k}} \cdot \mathcal{E}_{ek}(1; \rho_k)$		$\quad A, B, C, D, G_0, \ldots, G_{m-1} \in \mathbb{G}$
$\quad$ using $p_{i,k}$ from (6.1)	$\xleftarrow{x \leftarrow \{0,1\}^\lambda}$	$\quad f_{0,1}, \ldots, f_{m-1,n-1}, z_A, z_C, z \in \mathbb{Z}_p$
		$\quad \mathcal{V}_1(gk, \mathsf{crs}, B, x, A, B, C, \{f_{j,i}\}_{j=0,i=1}^{m-1,n-1}, z_A, z_C) = 1$
$\quad (f_{0,1}, \ldots, f_{m-1,n-1}, z_A, z_C) \leftarrow \mathcal{P}_1(x)$	$f_{0,1}, \ldots, f_{m-1,n-1}, z_A, z_C, z$	$\quad \forall j : f_{j,0} := x - \sum_{i=1}^{n-1} f_{j,i}$
$z := rx^m - \sum_{k=0}^{m-1} \rho_k x^k$	$\longrightarrow$	$\quad \prod_{i=0}^{N-1} c_i^{\prod_{j=1}^{m} f_{j,i_j}} \cdot \prod_{k=0}^{m-1} G_k^{-x^k} = \mathcal{E}_{ek}(1; z)$

Figure 6.3: Σ-protocol for a list $c_0, \ldots, c_{N-1}$ containing an encryption of 1

Lemma 7. *Let $m \geq 2$. The* Σ-protocol *in Fig. 6.3 is perfectly complete, SHVZK, $(m+1)$-special sound and has quasi-unique responses, under the DDH assumption.*

Proof. First we prove perfect completeness. By the perfect completeness of the Σ-protocol in Fig. 6.2 we have that $\mathcal{V}_1$ always accepts. Correctness of the last equation follows from the homomorphic property of ElGamal encryption since

$$
\begin{aligned}
\prod_{i=0}^{N-1} \quad c_i^{\prod_{j=0}^{m-1} f_{j,i_j}} \cdot \prod_{k=0}^{m-1} G_k^{-x^k} &= \prod_{i=0}^{N-1} c_i^{p_i(x)} \cdot \prod_{k=0}^{m-1} \left(\prod_{i=0}^{N-1} c_i^{p_{i,k}} \cdot \mathrm{Enc}(1;\rho_k) \right)^{-x^k} \\
&= \prod_{i=0}^{N-1} c_i^{p_i(x)} \cdot \prod_{k=0}^{m-1} \left(\prod_{i=0}^{N-1} c_i^{-p_{i,k}x^k} \cdot \mathrm{Enc}(1;-x^k\rho_k) \right) \\
&= \prod_{i=0}^{N-1} c_i^{p_i(x)} \cdot \prod_{i=0}^{N-1} c_i^{-\sum_{k=0}^{m-1} p_{i,k}x^k} \cdot \mathrm{Enc}\left(1;-\sum_{k=0}^{m-1} x^k\rho_k\right) \\
&= \prod_{i=0}^{N-1} c_i^{\delta_{\ell,i}x^m} \cdot \mathrm{Enc}\left(1;-\sum_{k=0}^{m-1} x^k\rho_k\right) \quad = \quad c_\ell^{x^m} \mathrm{Enc}\left(1;-\sum_{k=0}^{m-1} x^k\rho_k\right) \\
&= \mathrm{Enc}(1;rx^m) \cdot \mathrm{Enc}\left(1;-\textstyle\sum_{k=0}^{m-1} x^k\rho_k\right) \quad = \quad \mathrm{Enc}(1;z).
\end{aligned}
$$

We now describe a special honest verifier zero-knowledge simulator. It picks $B \leftarrow \mathbb{G}$ and $G_1,\dots,G_{m-1} \leftarrow \mathbb{G}^2$. It runs the SHVZK simulator for $\mathcal{P}_1$ to simulate A,C,D,z_A, $z_C,f_{0,1},\dots,f_{m-1,n-1}$ and computes the $f_{j,0}$'s accordingly. It picks $z \leftarrow \mathbb{Z}_p$ and computes G_0 from the last verification equation.

By the DDH assumption, $G_1,\dots,G_{m-1}$ in a real proof are indistinguishable from picking random pairs in $\mathbb{G}^2$ as in the simulation. We get independent, uniformly random B and z in both real proofs and simulations. By the perfect SHVZK of the simulator for $\mathcal{P}_1$ we also have the same distribution of $A,B,C,f_{j,i},z_A,z_C$ as in a real proof. Finally, G_0 is uniquely determined by the last verification equation in both real proofs and in simulations, so the two are indistinguishable. The last verification equation uniquely determines z, thus quasi-unique responses follow from the quasi-unique responses of the underlying Σ-protocol for $\mathcal{R}_1$.

Now we prove the protocol is $(m+1)$-special sound. Suppose an adversary can produce $(m+1)$ different accepting responses $(f_{j,i}^{(0)},z^{(0)}),\dots,(f_{j,i}^{(m)},z^{(m)})$ with respect to $m+1$ different challenges $x^{(0)},\dots,x^{(m)}$ and the same initial message. Assume that $m>1$. We use 3-special soundness of the Σ-protocol for $\mathcal{R}_1$ to extract openings $\delta_{\ell_j,i},a_{j,i}$ for B and A with $\delta_{\ell_j,i} \in \{0,1\}$ and $\sum_{i=0}^{n-1} \delta_{\ell_j,i} = 1$. The openings define $\ell := \sum_{j=0}^{m-1} \ell_j n^j$, where ℓ_j is the index of the only 1 in the sequence $(\delta_{\ell_j,0},\dots,\delta_{\ell_j,n-1})$. Following the proof, all answers satisfy $f_{j,i}^{(e)} = \delta_{\ell_j,i}x^{(e)} + a_{j,i}$ for $0 \leq e \leq m$, with overwhelming probability due to the binding property of the commitment scheme.

From $\delta_{\ell_j,i},a_{j,i}$ we can compute the polynomials $p_i(x) = \prod_{j=0}^{m-1}(\delta_{\ell_j,i}x + a_{j,i})$. Note that $p_\ell(x)$ is the only such polynomial with degree m in x. Based on this observation we rewrite the last verification equation as: $c_\ell^{x^m} \cdot \prod_{k=0}^{m-1} \tilde{G}_k^{x^k} = \mathrm{Enc}(1;z)$. Here the $\tilde{G}_k$ values are derived from the initial statement and $\delta_{\ell_j,i},a_{j,i}$. This equation holds for $x^{(0)},\dots,x^{(m)}$. Consider the Vandermonde matrix with the eth row given by $(1,x^{(e)},\dots,(x^{(e)})^m)$. The $x^{(e)}$ values are distinct, so the matrix is invertible. We can thus obtain a linear combination $\theta_0,\dots,\theta_n$ of the rows producing the vector $(0,\dots,0,1)$. We deduce $c_\ell = \prod_{e=0}^{m}\left(c_\ell^{(x^{(e)})^m} \cdot \prod_{k=0}^{m-1} \tilde{G}_k^{(x^{(e)})^k}\right)^{\theta_e} =$ Enc $\left(1;\sum_{e=0}^{m}\theta_e z^{(e)}\right)$, which provides an opening of c_ℓ to the plaintext 1 with randomness $r = \sum_{e=0}^{m}\theta_e z^{(e)}$. □

6.4.3 Correct Signature

We give in Fig. 6.4 a Σ-protocol for the relation

$$\mathcal{R}_{\text{sig}} = \left\{ \ ((pk,m,R,c),(sk,r)) : sk \in \mathbb{Z}_p \wedge vk = g^{sk} \in R \subset \mathbb{G}^* \wedge c = \mathcal{E}_{pk}(vk;r) \ \right\}$$

$$\begin{array}{lcl}
\underline{\mathcal{P}_{\text{sig}}(gk,\mathsf{crs},(pk,m,R,c),(r,sk))} & & \underline{\mathcal{V}_{\text{sig}}(gk,\mathsf{crs},(pk,m,R,c))} \\
s,t,r_a,r_b \leftarrow \mathbb{Z}_p;\ d \leftarrow \mathcal{E}_{ek}(g^{sk};t) & & \\
A \leftarrow \mathcal{E}_{pk}(g^s;r_a);\ B \leftarrow \mathcal{E}_{ek}(g^s;r_b) & & \text{Accept iff} \\
c_0 := d\cdot\mathcal{E}_{ek}(vk_0^{-1};0),\dots,c_{N-1} := d\cdot\mathcal{E}_{ek}(vk_{N-1}^{-1};0) & \xrightarrow{d,A,B,a_2} & \quad R \subset \mathbb{G};\ pk \in \mathbb{G}^*;\ d,A,B \in \mathbb{G}^2 \\
a_2 \leftarrow \mathcal{P}_2(gk,\mathsf{crs},(c_0,\dots,c_{N-1}),(\ell,t)) & \xleftarrow{x \leftarrow \{0,1\}^\lambda} & \quad z_s,z_a,z_b \in \mathbb{Z}_p \\
 & & \quad c^x A = \mathcal{E}_{pk}(g^{z_s};z_a) \\
z_s := sk\cdot x + s\ ;\ z_a := rx + r_a\ ;\ z_b := tx + r_b & & \quad d^x B = \mathcal{E}_{ek}(g^{z_s};z_b) \\
z_2 \leftarrow \mathcal{P}_2(x) & \xrightarrow{z_a,z_b,z_s,z_2} & \mathcal{V}_2(gk,\mathsf{crs},(c_0,\dots,c_{N-1}),a_2,x,z_2) = 1
\end{array}$$

Figure 6.4: Σ-protocol for $\mathcal{R}_{\text{sig}}$

Lemma 8. *The* Σ-protocol *in Fig. 6.4 is perfectly complete, SHVZK, $(m+1)$-special sound and has quasi-unique responses, under the DDH assumption.*

Proof. Perfect completeness follows by direct verification and the perfect completeness of $(\mathcal{P}_2,\mathcal{V}_2)$. The SHVZK simulator chooses $z_a,z_b,z_s \leftarrow \mathbb{Z}_p$ and $d \leftarrow \mathbb{G}^2$ at random and computes A,B from the verification equations. It runs the perfect SHVZK simulator for $\mathcal{P}_2$ to get a_2 and z_2. By the DDH assumption, d is indistinguishable from the ciphertexts in the real proof. In Both real proofs and simulations, z_a,z_b,z_t are uniformly random and uniquely determine A,B giving us SHVZK. Since the verification equations uniquely determine z_a,z_b and z_s and $(\mathcal{P}_2,\mathcal{V}_2)$ has quasi-unique responses, so must this protocol.

For $(m+1)$-special soundness, consider accepting answers z_a,z_b,z_s and z'_a,z'_b,z'_s to distinct challenges x and x'. From the first verification equation we get $c^{x-x'} = \mathcal{E}_{pk}(g^{z_s-z'_s};z_a-z'_a)$ giving $sk = \frac{z_s-z'_s}{x-x'}$ and $r = \frac{z_a-z'_a}{x-x'}$. The second verification equation gives $d^{x-x'} = \mathcal{E}_{pk}(g^{z_s-z'_s};z_b-z'_b)$ so d also encrypts g^{sk}. Finally, $(m+1)$-special soundness of the Σ-protocol for $\mathcal{R}_2$ then shows that $g^{sk} \in R$. □

Lemma 9. *Applying the Fiat-Shamir transformation to the protocol in Fig. 6.4 with* SoKSetup *as in Sect. 6.4 produces a signature of knowledge in the random oracle model, that is extractable and straightline f-extractable, under the DDH assumption.*

Proof. For simulatability, $\mathsf{SoKSimSetup}$ is identical to $\mathsf{SoKSetup}$ and $\mathsf{SoKSimSign}$ programs the random oracle to simulate proofs. Simulatability then follows from SHVZK.

For extractability we rely on rewinding, $(m+1)$-special soundness and quasi-unique responses, using [FKMV12]. For straightline f-extractability, we use the trapdoor τ to decrypt d in the proof transcript and obtain $vk = f(sk)$. $\square$

6.4.4 Correct Opening

Writing out the details of ElGamal encryption we get

$$\mathcal{R}_{\text{open}} = \left\{ \begin{array}{c} ((pk, c, vk), dk) : \\ dk \in \mathbb{Z}_p \wedge pk = g^{dk} \neq 1 \wedge c = (u, v) \in \mathbb{G}^2 \wedge vk \in \mathbb{G} \wedge (v/vk)^{dk} = u \end{array} \right\}$$

$$\begin{array}{lcl}
\underline{\mathcal{P}_{\text{open}}(gk, \mathsf{crs}, (pk, u, v, vk), dk)} & & \underline{\mathcal{V}_{\text{open}}(gk, \mathsf{crs}, (pk, u, v, vk))} \\
a \leftarrow \mathbb{Z}_p & A, B & \text{Accept if and only if} \\
A := g^a & \xrightarrow{\qquad\qquad} & \quad pk, u, v, vk \in \mathbb{G} \\
B := (v/vk)^a & x \leftarrow \mathbb{Z}_p & \quad pk \neq 1 \\
 & \xleftarrow{\qquad\qquad} & \quad pk^x A = g^z \\
z = dk \cdot x + a & z & \quad u^x B = (v/vk)^z \\
 & \xrightarrow{\qquad\qquad} &
\end{array}$$

Figure 6.5: Σ-protocol for correct decryption

Lemma 10. *The Σ-protocol in Fig. 6.5 is perfectly complete, perfect SHVZK, perfect 2-special sound and has unique responses. Also, applying the Fiat-Shamir transformation to it produces a NIZK proof.*

Proof. Perfect completeness follows by direct verification. The SHVZK simulator picks $z \leftarrow \mathbb{Z}_p$ and computes A, B from the verification equations. Both in real proofs and simulated proofs z is uniformly random and the verification equations determine the initial message uniquely, so we have perfect simulation. As the first verification equation determines z we have unique responses.

For 2-special soundness, let z and z' be accepting answers to distinct challenges x, x'. The first verification equation gives $pk^{x-x'} = g^{z-z'}$, so $dk = \frac{z-z'}{x-x'}$. The second gives $u^{x-x'} = (v/vk)^{z-z'}$, which shows $u = (v/vk)^{dk}$. Thus, vk was encrypted in (u, v). $\square$

6.4.5 Efficiency of our schemes

The efficiency of our schemes is determined by the signature of knowledge of Fig. 6.4. For a ring of $N = n^m$ users, this requires the prover to send $m+4$ ElGamal ciphertexts, 4

Scheme	Signature Size	Assumptions	Type
[ACJT00]	$3\mathbb{Z}_n^* + 4\mathbb{Z}$	Strong RSA	Group
[CL02]	$6\mathbb{Z}_n^* + 8\mathbb{Z}$	Strong RSA	Group
[DKNS04]	$12\mathbb{Z}_n^* + 12\mathbb{Z}$	Strong RSA	Ring/Group
[CG05]	$4\mathbb{Z}_n^* + 4\mathbb{Z}$	Strong RSA + DDH	Group
[GK14]	$(4\log_2 N)\mathbb{G} + (3\log_2 N + 1)\mathbb{Z}_p^*$	DDH	Ring
Ours	$(\log_2 N + 12)\mathbb{G} + \frac{1}{2}(3\log_2 N + 12)\mathbb{Z}_p^*$	DDH	Ring/Group

Table 6.1: Efficiency comparison between our instantiation and most efficient group and ring signatures based on RSA and/or DDH assumptions. $\mathbb{Z}_n^*, \mathbb{Z}, \mathbb{G}, \mathbb{Z}_p^*$ represent the size of RSA ring elements, integers, group elements and field elements, respectively.

Pedersen commitments and $m(n-1)+6$ elements of $\mathbb{Z}_p$. A full accountable ring signature includes an additional ElGamal encryption, i.e. $2m+12$ group elements and $m(n-1)+6$ field elements in total.

A signature can be computed using $mN + 3mn + 2m + 12$ group exponentiations as follows. Computing A, C and D in the bit proof requires $2mn+3$ exponentiations since exponentiation by $(1-2b_{i,j})$ amounts to a multiplication. By construction of c_i in Fig. 6.4, the first components of all c_i are identical in Fig. 6.3, so computing the first components of all G_k costs $2m$ exponentiations. The second components of all G_k require $mN+m$ exponentiations. We also need 9 exponentiations to compute B in Fig. 6.3, d, A and B in Fig. 6.4, and the ElGamal encryption of the public key.

Signatures can be verified using $N + 2mn + 2m + 15$ group exponentiations as follows: $N + 2m + 3$ exponentiations for the last verification equation in Fig. 6.3, $2mn+4$ for the equations in Fig. 6.2 and 8 for the first two verification equations in Fig. 6.4.

Our schemes can be instantiated over any group $\mathbb{G}$ where the DDH problem is computationally hard. Let us say the security parameter λ determines the bit size of the field elements as $|p| \approx \lambda$ bits and let $N = \text{poly}(\lambda)$. When group elements are much larger than field elements, say more than a factor λ, it is convenient to choose a large n. For instance, setting $n = \lambda + 1$ (in which case $m = O(1)$) the communication complexity amounts to a constant number of group elements and $m\lambda + 6$ field elements. When group and field elements are of roughly the same size, as can be the case for elliptic curve groups, our signatures have total size $m(n+1)+18$ elements. Setting $n = 4$ gives communication of roughly $5\log_4 N + 18 = \frac{5}{2}\log_2 N + 18$ elements.

In table 6.1, we compare our instantiation with prior work. Since our signatures require a logarithmic number of group elements, they enjoy shorter sizes than all previous signatures based on RSA and/or DDH assumptions, for sufficiently large security parameters. Indeed, a constant number of RSA ring elements typically requires $O(\lambda^3)$ bits whereas the elliptic curve instantiation of our protocol achieves $O(\lambda \log N)$ bit size. As long as λ is large enough our signatures will be shorter. Strictly speaking, we also require that $N \leq 2^{\lambda^2}$, but this is trivially satisfied for any meaningful value of λ (e.g. for 80 bit security, we can handle 2^{6400} users). Our signatures are also a factor 2.8 shorter than Groth and Kohlweiss ring signatures.

Chapter 7

Fully Dynamic Group Signatures

In the previous Chapter we focused on accountable ring signatures, giving a particularly efficient construction and instantiation. In Sect. 6.2.1, we also sketched how one could derive static group signatures from accountable group signatures. In this Chapter we will flesh out that construction and move our attention to fully dynamic group signatures, i.e. where users can join and leave the group at any time. Compared to accountable ring signatures the main difference is that the opener is fixed ahead of time and most importantly, that membership in the group is controlled centrally by a manager.

We start by taking a close look at existing security definitions for fully dynamic group signatures. We identify a number of shortcomings in existing security definitions and fill the gap by providing a formal rigorous security model for the primitive. Our model is general and is not tailored towards a specific design paradigm and can therefore, as we show, be used to argue about the security of different existing constructions following different design paradigms, including the one based on the scheme of Sect. 6.3. Our definitions are stringent and when possible incorporate protection against maliciously chosen keys.

In the process, we identify a subtle issue inherent to one design paradigm, where new members might try to implicate older ones by means of back-dated signatures. This is not captured by existing models. We propose some inexpensive fixes for some existing constructions to avoid the issue.

The research presented in this chapter was joint work with Jonathan Bootle, Andrea Cerulli, Essam Ghadafi and Jens Groth, and was published in [BCC^{+}16a]. The author contributed to the investigation in developing the definitions for fully dynamic group signatures (together with the other authors), developing the attacks against existing schemes (with Andrea Cerulli), and the real-world motivation for the backdating attacks.

7.1 Introduction

Group signatures, put forward by Chaum and van Heyst [CvH91], are a fundamental cryptographic primitive allowing a member of a group (administered by a designated manager) to anonymously sign messages on behalf of the group. In the case of a dispute, a designated tracing manager can revoke anonymity by revealing the signer. In many settings it is desirable to offer flexibility in joining and leaving the group. In static group signatures [BMW03], the group population is fixed once and for all at the setup phase. Partially dynamic group signatures [BSZ05, KY06] allow the enrolment of members in the group at any time but members cannot leave once they have joined. A challenging problem in group signatures is that of revocation, i.e. allowing removal of members from the group.

Shortcomings in Existing Models. While the security of the static and partially dynamic group settings has been rigorously formulated [BMW03, BSZ05, KY06, SSE$^+$12] and is now well understood, unfortunately, the security of their fully dynamic groups counterpart, which is more relevant to practice, has received less attention and is still lacking. In particular, the different design paradigms assume different (sometimes informal) models which do not necessarily generalise to other design approaches. This resulted in various models, the majority of which lack rigour. As a consequence, it can be difficult to compare the merits of the different constructions in terms of their security guarantees. Moreover, existing models place a large amount of trust in the different authorities and assume that their keys are generated honestly. This does not necessarily reflect scenarios arising in real applications. Furthermore, some existing models, as we show, fail to take into account some attacks which might be problematic for some applications of the primitive.

7.1.1 Motivation

"He Who Controls the Present Controls the Past", (George Orwell). Consider a scenario where the new leadership of an organisation or country wants to justify an unpopular policy (e.g. layoffs or removal of personal freedoms). A way to do that would be to back-date documents justifying the policy: thus, any animosity for the policy would be towards the old leadership. The new leadership is only maintaining the status quo.

Re-framing this in technical terms, we show that the notion of traceability in existing models following the revocation list approach, where the group manager periodically publishes information (i.e. revocation lists) about members excluded from the group, is too weak. In those models, the life of the scheme spans over different intervals (epochs) at the start of which the manager updates the revocation lists. Signatures in those models are bound to a specific epoch. It is vital for functionality that old valid signatures (i.e. those produced at earlier epochs by then-legitimate members) are accepted by the verification algorithm.

The issue we identify in those models is that they allow members who joined at recent epochs to sign messages w.r.t earlier epochs during which they were not members of the group. In a sense this may be considered as an attack against traceability, as those members were not in the group at that interval. Technically however, the scenario we describe is allowed by the model: the underlying issue is a gap between one's interpretation of group signatures and what the definition implies. Our expectation is that a signature bound to epoch τ was produced by a member of the group *at that time*. Current definitions however, allow for all past, current, and future members, as long as they were not revoked at time τ.

One may dismiss this attack as theoretical, since the old leadership might appeal to the opener. However, this might not always be possible: the opener may be controlled by the new leadership, or in a business setting an outgoing CEO or board member might be disinterested or disincentivised from pursuing the issue. Another possible criticism might be that the weakness is trivial, and would be silently fixed in any construction using the model.

We show that some state of the art constructions, as [NFHF09, LPY12b, LPY12a], are susceptible to this attack. Specifically, their membership certificates are not bound to the epochs of their issuance. As a result, a member can sign w.r.t. earlier epochs. We stress that neither the authors of those schemes claimed their schemes were immune against such an issue nor that their models were supposed to capture such an attack. Thus, such an issue might not be a problem for the applications they originally had in mind, but only in a more general case.

In order to have strong security guarantees from the different constructions, a rigorous and unified security model is necessary. This is the aim of this work as we believe this is a challenging problem that needs to be addressed, especially given the relevance of the primitive.

7.1.2 Contributions

We take a close look at the security definitions of fully dynamic group signatures. We provide a rigorous security model that generalises to the different design paradigms. In particular, our model covers both accumulator based and revocation list based approaches. Our model offers stringent security definitions and takes into account some attacks which were not considered by existing models. We give different flavors of our security definitions which capture both cases when the authorities' keys are adversarially generated and when such keys are honestly generated. We also show that our security definitions imply existing definitions for static and partially dynamic group signatures.

In the process, we identify a subtle difference between accumulator based and revocation list based approaches. Specifically, we identify a simple attack against traceability inherent to constructions following the latter approach and which is not captured by existing models. The attack allows a group member to sign w.r.t. intervals prior to her joining the group. The security notion modelled by current definitions prevents users from signing only if they are explicitly revoked.

To address this, our traceability definition models a stricter security notion: users are not authorised to sign unless they are non-revoked and are active (i.e. part of the group) at the time interval associated with the signature. We note this is already implied in the accumulator based approach: the signer proves membership in the current version of the group at the time of signing. We also propose a number of possible fixes to this issue in some existing schemes.

Finally, we show that a fully dynamic group signature scheme obtained from the generic construction of accountable ring signatures given in Sect. 6.3 is secure w.r.t. the stronger variant of our security definitions.

Outline. We present our model for fully dynamic group signatures in Section 7.2 and show that it implies existing definitions for static and partially dynamic group signatures. In Section 7.4 we analyse the security of three existing fully dynamic group signature schemes in our model.

7.2 Syntax and Security of Fully Dynamic Group Signatures

The parties involved in a Fully Dynamic Group Signature (FDGS) are: a group manager $\mathcal{GM}$ who authorises who can join the group; a tracing manager $\mathcal{TM}$ who can revoke anonymity by opening signatures; a set of users, each with a unique identity $\mathsf{i} \in \mathbb{N}$, who are potential group members. Users can join/leave the group at any time at the discretion of the group manager. We assume the group manager will regularly publish some information info_τ, associated with a distinct index τ (hereafter referred to as epoch). We assume that

τ can be recovered given info_τ and vice versa (i.e. there is bijection between the epochs and associated information). The information depicts changes to the group, for instance, it could include the current members of the group (as in accumulator-based constructions) or those who have been excluded from the group (as, e.g. required by constructions based on revocation lists). As in existing models, we assume that anyone can verify the well-formedness and authenticity of the published group information. By combining the group information for the current epoch with that of the preceding one, any party can identify the list of members who have been revoked at the current epoch. We assume that the epochs preserve the order in which their corresponding information was published. More precisely, for all $\tau_1, \tau_2 \in \mathcal{T}$ ($\mathcal{T}$ being the space of epochs) we require that $\tau_1 < \tau_2$ if info_{τ_1} preceded info_{τ_2}.

Unlike existing models, which assume honestly generated authorities' keys, we separate the generation of the authorities' keys from that of the public parameters, which might need to be generated by a trusted party. This allows us (where appropriate) to define stringent security that protects against adversarial authorities who might generate their keys maliciously. Our definitions can be adapted straight away to work for the weaker setting where authorities' keys are generated honestly as in existing models. For the sake of generality, we define the group key generation as a joint protocol between the group and tracing managers. Clearly, it is desirable in some cases to avoid such interaction and allow authorities to generate their own keys independently. This is a special case of our general definition where the protocol is regarded as two one-sided protocols.

An $\mathcal{FDGS}$ scheme consists of the following polynomial-time algorithms:

- $\mathrm{GSetup}(1^\lambda) \to \mathsf{crs}$: is run by a trusted third party. On input a security parameter λ, it outputs public parameters crs. The algorithm also initialises the registration table reg.

- $\langle \mathsf{GKGen}_{\mathcal{GM}}(\mathsf{crs}), \mathsf{GKGen}_{\mathcal{TM}}(\mathsf{crs}) \rangle$: is an interactive protocol between algorithms $\mathsf{GKGen}_{\mathcal{GM}}$ and $\mathsf{GKGen}_{\mathcal{TM}}$ run by $\mathcal{GM}$ and $\mathcal{TM}$, respectively, to generate their respective private keys as well as the rest of the group public key gpk. The input to both algorithms is the public parameters crs. If completed successfully, the private output of $\mathsf{GKGen}_{\mathcal{GM}}$ is a secret manager key msk, whereas its public output is a public key mpk, and the initial group information info. The private output of $\mathsf{GKGen}_{\mathcal{TM}}$ is the secret tracing key tsk, whereas its public output is a public key tpk. The group public key is then set to $\mathsf{gpk} := (\mathsf{crs}, \mathsf{mpk}, \mathsf{tpk})$.

- $\mathsf{UKGen}(1^\lambda) \to (\mathbf{usk}[\mathsf{i}], \mathbf{upk}[\mathsf{i}])$: outputs a secret/public key pair ($\mathbf{usk}[\mathsf{i}]$, $\mathbf{upk}[\mathsf{i}]$) for user i. We assume the public key table **upk** to be publicly available (possibly via PKI) so that anyone can get authentic copies of it.

- $\langle \mathsf{Join}(\mathsf{info}_{\tau_{\text{current}}}, \mathsf{gpk}, \mathsf{i}, \mathbf{usk}[\mathsf{i}]), \mathsf{Issue}(\mathsf{info}_{\tau_{\text{current}}}, \mathsf{msk}, \mathsf{i}, \mathbf{upk}[\mathsf{i}]) \rangle$: is an interactive protocol between a user i (who has already obtained a personal key pair, i.e. ran the UKGen algorithm) and the group manager $\mathcal{GM}$. Upon successful completion, i becomes a member of the group. The final state of the Issue algorithm is stored in the registration table at index i (i.e. $\mathsf{reg}[\mathsf{i}]$), whereas that of the Join algorithm is stored in $\mathbf{gsk}[\mathsf{i}]$. The epoch τ_{current} is part of the output of both parties.

  We assume that the protocol takes place over a secure (i.e. private and authentic) channel. The protocol is initiated by calling Join. The manager may update the group information after running this protocol. The registration table reg stores additional information used by the group manager and the tracing manager for updating and tracing, depending on the scheme specifics.

- $\mathsf{UpdateGroup}(\mathsf{gpk}, \mathsf{msk}, \mathsf{info}_{\tau_{\text{current}}}, \mathcal{S}, \mathsf{reg}) \to \mathsf{info}_{\tau_{\text{new}}}$: is run by the group manager to update the group information while also advancing the epoch. It takes as input the group manager's secret key msk, a (possibly empty) set $\mathcal{S}$ of active members to be removed from the group and the registration table reg, it outputs a new group information $\mathsf{info}_{\tau_{\text{new}}}$ and might also update the registration table reg. If there has been no changes to the group information, the algorithm returns $\perp$ to indicate that no new information has been issued. The algorithm aborts if any $\mathsf{i} \in \mathcal{S}$ has not run the join protocol.

- $\mathsf{Sign}(\mathsf{gpk}, \mathbf{gsk}[\mathsf{i}], \mathsf{info}_\tau, m) \to \Sigma$: on input the group public key gpk, a user's group signing key $\mathbf{gsk}[\mathsf{i}]$, the group information info_τ at epoch τ, and a message m, outputs a group signature Σ on m by the group member i. If the user owning $\mathbf{gsk}[\mathsf{i}]$ is not an active member of the group at epoch τ, the algorithm returns $\perp$.

- $\mathsf{Verify}(\mathsf{gpk}, \mathsf{info}_\tau, m, \Sigma) \to 1/0$: is a deterministic algorithm checking whether Σ is a valid group signature on m at epoch τ and outputs a bit accordingly.

- $\mathsf{Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_\tau, \mathsf{reg}, m, \Sigma) \to (\mathsf{i}, \pi_{\text{Trace}})$: is a deterministic algorithm which is run by the tracing manager. It returns an identity $\mathsf{i} > 0$ of the group member who produced Σ plus a proof π_{Trace} attesting to this fact. If the algorithm is unable to trace the signature to a particular group member, it returns $(0, \pi_{\text{Trace}})$ to indicate that it could not attribute the signature.

- $\mathsf{Judge}(\mathsf{gpk}, \mathsf{i}, \mathsf{info}_\tau, \pi_{\text{Trace}}, \mathbf{upk}[\mathsf{i}], m, \Sigma) \rightarrow 1/0$: is a deterministic algorithm which on input the group public key gpk, a user identity i, the group information at epoch τ, a tracing proof π_{Trace}, the user's public key $\mathbf{upk}[\mathsf{i}]$ (which is $\perp$ if it does not exist), a message m, and a signature Σ, outputs 1 if π_{Trace} is a valid proof that i produced Σ, and outputs 0 otherwise.

ADDITIONAL ALGORITHM. We will also use the following polynomial-time algorithm which is only used in the security games to ease composition.

$\mathsf{IsActive}(\mathsf{info}_\tau, \mathsf{reg}, \mathsf{i}) \rightarrow 1/0$: returns 1 if the user i is an active member of the group at epoch τ and 0 otherwise.

7.2.1 Security of Fully Dynamic Group Signatures

The security requirements of a fully dynamic group signature are: *correctness*, *anonymity*, *non-frameability*, *traceability* and *tracing soundness*. To define those requirements, we use a set of games in which the adversary has access to a set of oracles. The following global lists are maintained: HUL is a list of honest users; CUL is a list of corrupt users whose personal secret keys have been chosen by the adversary; BUL is a list of bad users whose personal and group signing keys have been revealed to the adversary; SL is a list of signatures obtained from the Sign oracle; CL is a list of challenge signatures obtained from the challenge oracle.

The details of the following oracles are given in Fig. 7.1.

$\mathsf{UToM}(\mathsf{i})$ adds an honest user i to the group at the current epoch.

$\mathsf{CI}(\mathsf{i}, \mathsf{pk})$ creates a new corrupt user whose public key $\mathbf{upk}[\mathsf{i}]$ is chosen by the adversary. This is called in preparation for calling the SndToM oracle.

$\mathsf{SndToM}(\mathsf{i}, M_{\mathsf{in}})$ used to engage in the Join-Issue protocol with the honest, Issue-executing group manager.

$\mathsf{SndToU}(\mathsf{i}, M_{\mathsf{in}})$ used to engage in the Join-Issue protocol with an honest, Join-executing user i on behalf of the corrupt group manager.

```
$\underline{\mathsf{UToM}(\mathsf{i})}$
  • If $\mathsf{i} \in \mathsf{HUL} \cup \mathsf{CUL}$ Then Return $\perp$.
  • $(\mathbf{usk}[\mathsf{i}], \mathbf{upk}[\mathsf{i}]) \leftarrow \mathsf{UKGen}(1^\lambda)$.
  • $\mathsf{HUL} := \mathsf{HUL} \cup \{\mathsf{i}\}$, $\mathbf{gsk}[\mathsf{i}] := \perp$, $\mathsf{dec}^{\mathsf{i}}_{\mathsf{Issue}} := \mathsf{cont}$.
  • $\mathsf{st}^{\mathsf{i}}_{\mathsf{Join}} := (\tau_{\mathrm{current}}, \mathsf{gpk}, \mathsf{i}, \mathbf{usk}[\mathsf{i}])$.
  • $\mathsf{st}^{\mathsf{i}}_{\mathsf{Issue}} := (\tau_{\mathrm{current}}, \mathsf{msk}, \mathsf{i}, \mathbf{upk}[\mathsf{i}])$.
  • $(\mathsf{st}^{\mathsf{i}}_{\mathsf{Join}}, M_{\mathsf{Issue}}, \mathsf{dec}^{\mathsf{i}}_{\mathsf{Join}}) \leftarrow \mathsf{Join}(\mathsf{st}^{\mathsf{i}}_{\mathsf{Join}}, \perp)$.
  • While $(\mathsf{dec}^{\mathsf{i}}_{\mathsf{Issue}} = \mathsf{cont}$ and $\mathsf{dec}^{\mathsf{i}}_{\mathsf{Join}} = \mathsf{cont})$ Do
      ◦ $(\mathsf{st}^{\mathsf{i}}_{\mathsf{Issue}}, M_{\mathsf{Join}}, \mathsf{dec}^{\mathsf{i}}_{\mathsf{Issue}}) \leftarrow \mathsf{Issue}(\mathsf{st}^{\mathsf{i}}_{\mathsf{Issue}}, M_{\mathsf{Issue}})$.
      ◦ $(\mathsf{st}^{\mathsf{i}}_{\mathsf{Join}}, M_{\mathsf{Issue}}, \mathsf{dec}^{\mathsf{i}}_{\mathsf{Join}}) \leftarrow \mathsf{Join}(\mathsf{st}^{\mathsf{i}}_{\mathsf{Join}}, M_{\mathsf{Join}})$.
  • If $\mathsf{dec}^{\mathsf{i}}_{\mathsf{Issue}} = \mathsf{accept}$ Then $\mathsf{reg}[\mathsf{i}] := \mathsf{st}^{\mathsf{i}}_{\mathsf{Issue}}$.
  • If $\mathsf{dec}^{\mathsf{i}}_{\mathsf{Join}} = \mathsf{accept}$ Then $\mathbf{gsk}[\mathsf{i}] := \mathsf{st}^{\mathsf{i}}_{\mathsf{Join}}$.
  • Return $\mathbf{upk}[\mathsf{i}]$.

$\underline{\mathsf{SndToU}(\mathsf{i}, M_{\mathsf{in}})}$
  • If $\mathsf{i} \in \mathsf{CUL} \cup \mathsf{BUL}$ Then Return $\perp$.
  • If $\mathsf{i} \notin \mathsf{HUL}$ Then
      ◦ $\mathsf{HUL} := \mathsf{HUL} \cup \{\mathsf{i}\}$.
      ◦ $(\mathbf{usk}[\mathsf{i}], \mathbf{upk}[\mathsf{i}]) \leftarrow \mathsf{UKGen}(1^\lambda)$.
      ◦ $\mathbf{gsk}[\mathsf{i}] := \perp$, $M_{\mathsf{in}} := \perp$.
  • If $\mathsf{dec}^{\mathsf{i}}_{\mathsf{Join}} \neq \mathsf{cont}$ Then Return $\perp$.
  • If $\mathsf{st}^{\mathsf{i}}_{\mathsf{Join}}$ is undefined
      ◦ $\mathsf{st}^{\mathsf{i}}_{\mathsf{Join}} := (\tau_{\mathrm{current}}, \mathsf{gpk}, \mathsf{i}, \mathbf{usk}[\mathsf{i}])$.
  • $(\mathsf{st}^{\mathsf{i}}_{\mathsf{Join}}, M_{\mathsf{out}}, \mathsf{dec}^{\mathsf{i}}_{\mathsf{Join}}) \leftarrow \mathsf{Join}(\mathsf{st}^{\mathsf{i}}_{\mathsf{Join}}, M_{\mathsf{in}})$
  • If $\mathsf{dec}^{\mathsf{i}}_{\mathsf{Join}} = \mathsf{accept}$ Then $\mathbf{gsk}[\mathsf{i}] := \mathsf{st}^{\mathsf{i}}_{\mathsf{Join}}$.
  • Return $(M_{\mathsf{out}}, \mathsf{dec}^{\mathsf{i}}_{\mathsf{Join}})$.

$\underline{\mathsf{Trace}(m, \Sigma, \mathsf{info}_\tau)}$
  • Return $(\perp, \perp)$ if $\mathsf{Verify}(\mathsf{gpk}, \mathsf{info}_\tau, m, \Sigma) = 0$.
  • Return $(\perp, \perp)$ if $(m, \Sigma, \tau) \in \mathsf{CL}$.
  • Return $\mathsf{Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_\tau, \mathsf{reg}, m, \Sigma)$.

$\underline{\mathsf{ReadReg}(\mathsf{i})}$
  • Return $\mathsf{reg}[\mathsf{i}]$.

$\underline{\mathsf{Reveal}(\mathsf{i})}$
  • Return $\perp$ if $\mathsf{i} \notin \mathsf{HUL} \setminus (\mathsf{CUL} \cup \mathsf{BUL})$.
  • $\mathsf{BUL} := \mathsf{BUL} \cup \{\mathsf{i}\}$.
  • Return $(\mathbf{usk}[\mathsf{i}], \mathbf{gsk}[\mathsf{i}])$.

$\underline{\mathsf{CI}(\mathsf{i}, \mathsf{pk})}$
  • Return $\perp$ if $\mathsf{i} \in \mathsf{HUL} \cup \mathsf{CUL}$.
  • $\mathsf{CUL} := \mathsf{CUL} \cup \{\mathsf{i}\}$.
  • $\mathbf{upk}[\mathsf{i}] := \mathsf{pk}$, $\mathsf{dec}^{\mathsf{i}}_{\mathsf{Issue}} := \mathsf{cont}$.
  • Return accept.

$\underline{\mathsf{SndToM}(\mathsf{i}, M_{\mathsf{in}})}$
  • Return $\perp$ if $\mathsf{i} \notin \mathsf{CUL}$.
  • Return $\perp$ if $\mathsf{dec}^{\mathsf{i}}_{\mathsf{Issue}} \neq \mathsf{cont}$.
  • $\mathsf{st}^{\mathsf{i}}_{\mathsf{Issue}} := (\tau_{\mathrm{current}}, \mathsf{msk}, \mathsf{i}, \mathbf{upk}[\mathsf{i}])$.
  • $(\mathsf{st}^{\mathsf{i}}_{\mathsf{Issue}}, M_{\mathsf{out}}, \mathsf{dec}^{\mathsf{i}}_{\mathsf{Issue}}) \leftarrow \mathsf{Issue}(\mathsf{st}^{\mathsf{i}}_{\mathsf{Issue}}, M_{\mathsf{in}})$.
  • If $\mathsf{dec}^{\mathsf{i}}_{\mathsf{Issue}} = \mathsf{accept}$ Then $\mathsf{reg}[\mathsf{i}] := \mathsf{st}^{\mathsf{i}}_{\mathsf{Issue}}$.
  • Return $(M_{\mathsf{out}}, \mathsf{dec}^{\mathsf{i}}_{\mathsf{Issue}})$.

$\underline{\mathsf{Sign}(\mathsf{i}, m, \tau)}$
  • Return $\perp$ if $\mathsf{i} \notin \mathsf{HUL}$ or $\mathbf{gsk}[\mathsf{i}] = \perp$ or $\mathsf{info}_\tau = \perp$.
  • Return $\perp$ if $\mathsf{IsActive}(\mathsf{info}_\tau, \mathsf{reg}, \mathsf{i}) = 0$.
  • $\Sigma \leftarrow \mathsf{Sign}(\mathsf{gpk}, \mathbf{gsk}[\mathsf{i}], \mathsf{info}_\tau, m)$.
  • $\mathsf{SL} := \mathsf{SL} \cup \{(\mathsf{i}, m, \Sigma, \tau)\}$.
  • Return $\Sigma$.

$\underline{\mathsf{Chal}_b(\mathsf{info}_\tau, \mathsf{i}_0, \mathsf{i}_1, m)}$
  • Return $\perp$ if $\mathsf{i}_0 \notin \mathsf{HUL}$ or $\mathsf{i}_1 \notin \mathsf{HUL}$.
  • Return $\perp$ if $\exists b \in \{0,1\}$ s.t. $\mathbf{gsk}[\mathsf{i}_b] = \perp$.
  • Return $\perp$ if $\exists b \in \{0,1\}$ s.t. $\mathsf{IsActive}(\mathsf{info}_\tau, \mathsf{reg}, \mathsf{i}_b) = 0$.
  • $\Sigma \leftarrow \mathsf{Sign}(\mathsf{gpk}, \mathbf{gsk}[\mathsf{i}_b], \mathsf{info}_\tau, m)$.
  • $\mathsf{CL} := \mathsf{CL} \cup \{(m, \Sigma, \tau)\}$.
  • Return $\Sigma$.

$\underline{\mathsf{ModifyReg}(\mathsf{i}, val)}$
  • $\mathsf{reg}[\mathsf{i}] := val$.

$\underline{\mathsf{UpdateGroup}(\mathcal{S})}$
  • Return $\mathsf{UpdateGroup}(\mathsf{gpk}, \mathsf{msk}, \mathsf{info}_{\tau_{\mathrm{current}}}, \mathcal{S}, \mathsf{reg})$.
```

Figure 7.1: Details of the oracles used in the security games

$\mathsf{ReadReg}(\mathsf{i})$ returns the registration information $\mathsf{reg}[\mathsf{i}]$ of user i.

$\mathsf{ModifyReg}(\mathsf{i}, val)$ modifies the entry $\mathsf{reg}[\mathsf{i}]$, setting $\mathsf{reg}[\mathsf{i}] := val$. For brevity we will assume $\mathsf{ModifyReg}$ also provides the functionality of $\mathsf{ReadReg}$.

$\mathsf{Reveal}(\mathsf{i})$ returns the personal secret key $\mathbf{usk}[\mathsf{i}]$ and group signing key $\mathbf{gsk}[\mathsf{i}]$ of group member i.

$\mathsf{Sign}(\mathsf{i}, m, \tau)$ returns a signature on the message m by the group member i for epoch τ assuming the corresponding group information info_τ is defined.

$\mathsf{Chal}_b(\mathsf{info}_\tau, \mathsf{i}_0, \mathsf{i}_1, m)$ is a left-right oracle for defining anonymity. The adversary chooses an epoch τ, the group information info_τ, two identities $(\mathsf{i}_0, \mathsf{i}_1)$, and a message m and receives a group signature by member i_b for $b \leftarrow \{0,1\}$ for the chosen epoch. It is required that both challenge users are active members at epoch τ. The adversary can only call this oracle once.

$\mathsf{Trace}(m, \Sigma, \mathsf{info}_\tau)$ returns the identity of the signer of the signature Σ on m w.r.t. info_τ if the signature was not obtained from the Chal_b oracle.

$\mathsf{UpdateGroup}(\mathcal{S})$ allows the adversary to update the group. $\mathcal{S}$ here is the set of the active members to be removed from the group.

The following security requirements are defined by the games in Fig. 7.2.

Correctness. This requirement guarantees that signatures produced by honest, non-revoked users are accepted by the Verify algorithm and that the honest tracing manager can identify the signer of such signatures. In addition, the Judge algorithm accepts the tracing manager's decision.

Formally, an $\mathcal{FDGS}$ scheme is *(perfectly) correct* if for all $\lambda \in \mathbb{N}$, the advantage

$$\mathsf{Adv}^{\mathrm{Corr}}_{\mathcal{FDGS},\mathcal{A}}(\lambda) := \Pr[\mathbf{Exp}^{\mathrm{Corr}}_{\mathcal{FDGS},\mathcal{A}}(\lambda) = 1]$$

is negligible (in λ) for all adversaries $\mathcal{A}$.

Note that the above definition of (perfect) correctness protects against even unbounded adversaries. If computational correctness suffices, i.e. when we consider correctness only against computationally-bounded adversaries, we can drop the last three lines from the correctness game in Fig. 7.2. Computational correctness of the Trace and Judge algorithms is implied by the other requirements.

(Full) Anonymity. This requires that signatures do not reveal the identity of the group member who produced them. In the game, the adversary, $\mathcal{A}$, can corrupt any user and fully corrupt the group manager by choosing her key. We require that both challenge users are active members of the group at the chosen epoch. Also, note that a Trace query on the challenge signature will fail.

As $\mathcal{A}$ can learn the personal secret and group signing keys of any user, including the challenge users, our definition captures full key exposure attacks.

The adversary chooses an epoch, the group information for that epoch, a message and two group members and gets a signature by either member and wins if she correctly guesses

```
Experiment: $\mathbf{Exp}^{\mathrm{Corr}}_{\mathcal{FDGS},\mathcal{A}}(\lambda)$
– $\mathsf{crs} \leftarrow \mathsf{GSetup}(1^{\lambda})$; $\mathsf{HUL} := \emptyset$.
– $\Big((\mathsf{msk},\mathsf{mpk},\mathsf{info}),(\mathsf{tsk},\mathsf{tpk})\Big) \leftarrow \langle\langle \mathsf{GKGen}_{\mathcal{GM}}(\mathsf{crs}),\mathsf{GKGen}_{\mathcal{TM}}(\mathsf{crs})\rangle\rangle$.
– $\mathsf{gpk} := (\mathsf{crs},\mathsf{mpk},\mathsf{tpk})$.
– $\Big(\mathsf{i},m,\tau\Big) \leftarrow \mathcal{A}^{\mathsf{UToM},\mathsf{ReadReg},\mathsf{UpdateGroup}}\Big(\mathsf{gpk},\mathsf{info}\Big)$.
– If $\mathsf{i} \notin \mathsf{HUL}$ or $\mathbf{gsk}[\mathsf{i}] = \perp$ or $\mathsf{info}_{\tau} = \perp$ or $\mathsf{IsActive}(\mathsf{info}_{\tau},\mathsf{reg},\mathsf{i}) = 0$ Then Return 0.
– $\Sigma \leftarrow \mathsf{Sign}(\mathsf{gpk},\mathbf{gsk}[\mathsf{i}],\mathsf{info}_{\tau},m)$.
– If $\mathsf{Verify}(\mathsf{gpk},\mathsf{info}_{\tau},m,\Sigma) = 0$ Then Return 1.
– $(\mathsf{i}^*,\pi_{\mathrm{Trace}}) \leftarrow \mathsf{Trace}(\mathsf{gpk},\mathsf{tsk},\mathsf{info}_{\tau},\mathsf{reg},m,\Sigma)$.
– If $\mathsf{i} \neq \mathsf{i}^*$ Then Return 1.
– If $\mathsf{Judge}(\mathsf{gpk},\mathsf{i},\mathsf{info}_{\tau},\pi_{\mathrm{Trace}},\mathbf{upk}[\mathsf{i}],m,\Sigma) = 0$ Then Return 1, Else Return 0.

Experiment: $\mathbf{Exp}^{\mathrm{Anon}\text{-}b}_{\mathcal{FDGS},\mathcal{A}}(\lambda)$
– $\mathsf{crs} \leftarrow \mathsf{GSetup}(1^{\lambda})$; $\mathsf{HUL},\mathsf{CUL},\mathsf{BUL},\mathsf{SL},\mathsf{CL} := \emptyset$.
– $\Big(\mathsf{st}_{\mathsf{init}},\mathsf{msk},\mathsf{mpk},\mathsf{info}\Big) \leftarrow \mathcal{A}^{\langle\cdot,\mathsf{GKGen}_{\mathcal{TM}}(\mathsf{crs})\rangle}(\mathsf{init}:\mathsf{crs})$.
– Return 0 if $\mathsf{GKGen}_{\mathcal{TM}}$ did not accept or $\mathcal{A}$'s output is not well-formed.
– Parse the output of $\mathsf{GKGen}_{\mathcal{TM}}$ as $(\mathsf{tsk},\mathsf{tpk})$ and set $\mathsf{gpk} := (\mathsf{crs},\mathsf{mpk},\mathsf{tpk})$.
– $b^* \leftarrow \mathcal{A}^{\mathsf{UToM},\mathsf{CI},\mathsf{SndToU},\mathsf{Reveal},\mathsf{Trace},\mathsf{ModifyReg},\mathsf{Chal}_b}\Big(\mathsf{play}:\mathsf{st}_{\mathsf{init}},\mathsf{gpk}\Big)$.
– Return $b^*$.

Experiment: $\mathbf{Exp}^{\mathrm{Non\text{-}Frame}}_{\mathcal{FDGS},\mathcal{A}}(\lambda)$
– $\mathsf{crs} \leftarrow \mathsf{GSetup}(1^{\lambda})$; $\mathsf{HUL},\mathsf{CUL},\mathsf{BUL},\mathsf{SL} := \emptyset$.
– $(\mathsf{st}_{\mathsf{init}},\mathsf{info},\mathsf{msk},\mathsf{mpk},\mathsf{tsk},\mathsf{tpk}) \leftarrow \mathcal{A}(\mathsf{init}:\mathsf{crs})$.
– Return 0 if $\mathcal{A}$'s output is not well-formed otherwise set $\mathsf{gpk} := (\mathsf{crs},\mathsf{mpk},\mathsf{tpk})$.
– $\Big(m,\Sigma,\mathsf{i},\pi_{\mathrm{Trace}},\mathsf{info}_{\tau}\Big) \leftarrow \mathcal{A}^{\mathsf{CI},\mathsf{SndToU},\mathsf{Reveal},\mathsf{Sign},\mathsf{ModifyReg}}\Big(\mathsf{play}:\mathsf{st}_{\mathsf{init}},\mathsf{gpk}\Big)$.
– If $\mathsf{Verify}(\mathsf{gpk},\mathsf{info}_{\tau},m,\Sigma) = 0$ Then Return 0.
– If $\mathsf{Judge}(\mathsf{gpk},\mathsf{i},\mathsf{info}_{\tau},\pi_{\mathrm{Trace}},\mathbf{upk}[\mathsf{i}],m,\Sigma) = 0$ Then Return 0.
– If $\mathsf{i} \notin \mathsf{HUL} \setminus \mathsf{BUL}$ or $(\mathsf{i},m,\Sigma,\tau) \in \mathsf{SL}$ Then Return 0 Else Return 1.

Experiment: $\mathbf{Exp}^{\mathrm{Trace}}_{\mathcal{FDGS},\mathcal{A}}(\lambda)$
– $\mathsf{crs} \leftarrow \mathsf{GSetup}(1^{\lambda})$; $\mathsf{HUL},\mathsf{CUL},\mathsf{BUL},\mathsf{SL} := \emptyset$.
– $\Big(\mathsf{st}_{\mathsf{init}},\mathsf{tsk},\mathsf{tpk}\Big) \leftarrow \mathcal{A}^{\langle\mathsf{GKGen}_{\mathcal{GM}}(\mathsf{crs}),\cdot\rangle}(\mathsf{init}:\mathsf{crs})$.
– Return 0 if $\mathsf{GKGen}_{\mathcal{GM}}$ did not accept or $\mathcal{A}$'s output is not well-formed.
– Parse the output of $\mathsf{GKGen}_{\mathcal{GM}}$ as $(\mathsf{msk},\mathsf{mpk},\mathsf{info})$. Set $\mathsf{gpk} := (\mathsf{crs},\mathsf{mpk},\mathsf{tpk})$.
– $\Big(m,\Sigma,\tau\Big) \leftarrow \mathcal{A}^{\mathsf{UToM},\mathsf{CI},\mathsf{SndToM},\mathsf{Reveal},\mathsf{Sign},\mathsf{ReadReg},\mathsf{UpdateGroup}}\Big(\mathsf{play}:\mathsf{st}_{\mathsf{init}},\mathsf{gpk},\mathsf{info}\Big)$.
– If $\mathsf{Verify}(\mathsf{gpk},\mathsf{info}_{\tau},m,\Sigma) = 0$ Then Return 0.
– $(\mathsf{i},\pi_{\mathrm{Trace}}) \leftarrow \mathsf{Trace}(\mathsf{gpk},\mathsf{tsk},\mathsf{info}_{\tau},\mathsf{reg},m,\Sigma)$.
– If $\mathsf{IsActive}(\mathsf{info}_{\tau},\mathsf{reg},\mathsf{i}) = 0$ Then Return 1.
– If $\mathsf{i} = 0$ or $\mathsf{Judge}(\mathsf{gpk},\mathsf{i},\mathsf{info}_{\tau},\pi_{\mathrm{Trace}},\mathbf{upk}[\mathsf{i}],m,\Sigma) = 0$ Then Return 1 Else Return 0.

Experiment: $\mathbf{Exp}^{\mathrm{Trace\text{-}Sound}}_{\mathcal{FDGS},\mathcal{A}}(\lambda)$
– $\mathsf{crs} \leftarrow \mathsf{GSetup}(1^{\lambda})$; $\mathsf{CUL} := \emptyset$.
– $(\mathsf{st}_{\mathsf{init}},\mathsf{info},\mathsf{msk},\mathsf{mpk},\mathsf{tsk},\mathsf{tpk}) \leftarrow \mathcal{A}(\mathsf{init}:\mathsf{crs})$.
– Return 0 if $\mathcal{A}$'s output is not well-formed otherwise set $\mathsf{gpk} := (\mathsf{crs},\mathsf{mpk},\mathsf{tpk})$.
– $\Big(m,\Sigma,\{\mathsf{i}_i,\pi_{\mathrm{Trace}_i}\}_{i=1}^{2},\mathsf{info}_{\tau}\Big) \leftarrow \mathcal{A}^{\mathsf{CI},\mathsf{ModifyReg}}\Big(\mathsf{play}:\mathsf{st}_{\mathsf{init}},\mathsf{gpk}\Big)$.
– If $\mathsf{Verify}(\mathsf{gpk},\mathsf{info}_{\tau},m,\Sigma) = 0$ Then Return 0.
– If $\mathsf{i}_1 = \mathsf{i}_2$ or $\mathsf{i}_1 = \perp$ or $\mathsf{i}_2 = \perp$ Then Return 0.
– If $\exists i \in \{1,2\}$ s.t. $\mathsf{Judge}(\mathsf{gpk},\mathsf{i}_i,\mathsf{info}_{\tau},\pi_{\mathrm{Trace}_i},\mathbf{upk}[\mathsf{i}_i],m,\Sigma) = 0$ Then Return 0.
– Return 1.
```

Figure 7.2: Security games for fully dynamic group signatures

the member. Without loss in generality, we allow the adversary a single call to the challenge oracle. A hybrid argument (similar to that used in [BSZ05]) can be used to prove that this is sufficient.

Formally, an $\mathcal{FDGS}$ scheme is *(fully) anonymous* if for all $\lambda \in \mathbb{N}$, the advantage $\mathsf{Adv}^{\mathrm{Anon}}_{\mathcal{FDGS},\mathcal{A}}(\lambda)$ is negligible (in λ) for all PPT adversaries $\mathcal{A}$, where

$$\mathsf{Adv}^{\mathrm{Anon}}_{\mathcal{FDGS},\mathcal{A}}(\lambda) := \left|\Pr[\mathbf{Exp}^{\mathrm{Anon\text{-}0}}_{\mathcal{FDGS},\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{\mathrm{Anon\text{-}1}}_{\mathcal{FDGS},\mathcal{A}}(\lambda) = 1]\right|.$$

Non-Frameability. This ensures that even if the rest of the group as well as the tracing and group managers are fully corrupt, they cannot produce a signature that can be attributed to an honest member who did not produce it.

In the game, the adversary can fully corrupt both the group and tracing managers. She even chooses the keys of both managers. Thus, our definition is stronger than existing models. We just require that the framed member is honest.

Formally, an $\mathcal{FDGS}$ scheme is *non-frameable* if for all $\lambda \in \mathbb{N}$, the advantage

$$\mathsf{Adv}^{\mathrm{Non\text{-}Frame}}_{\mathcal{FDGS},\mathcal{A}}(\lambda) := \Pr[\mathbf{Exp}^{\mathrm{Non\text{-}Frame}}_{\mathcal{FDGS},\mathcal{A}}(\lambda) = 1]$$

is negligible (in λ) for all PPT adversaries $\mathcal{A}$.

Remark 1. *In the game variant we give in Fig. 7.2, we allow the adversary to generate the tracing manager's key herself. While, as we show later, there are schemes which satisfy this strong variant of the definition, such definition might be too strong to be satisfied by some existing schemes. A weaker variant of the definition is where the tracing key is generated by the challenger rather than the adversary. This requires replacing lines 2-4 in the game in Fig. 7.2 by the following:*

- $(\mathsf{st_{init}}, \mathsf{info}, \mathsf{msk}, \mathsf{mpk}) \leftarrow \mathcal{A}^{\langle\cdot, \mathsf{GKGen}_{\mathcal{TM}}(\mathsf{crs})\rangle}(\mathsf{init} : \mathsf{crs})$.
- Return 0 if $\mathcal{A}$'s output is not well-formed or $\mathsf{GKGen}_{\mathcal{TM}}$ did not accept.
- Let $(\mathsf{tsk}, \mathsf{tpk})$ be the output of $\mathsf{GKGen}_{\mathcal{TM}}$. Set $\mathsf{gpk} := (\mathsf{crs}, \mathsf{mpk}, \mathsf{tpk})$.
- $\left(m, \Sigma, \mathsf{i}, \pi_{\mathrm{Trace}}, \mathsf{info}_{\tau}\right) \leftarrow \mathcal{A}^{\mathsf{CI}, \mathsf{SndToU}, \mathrm{Reveal}, \mathrm{Sign}, \mathrm{ModifyReg}}\left(\mathsf{play} : \mathsf{st_{init}}, \mathsf{gpk}, \mathsf{tsk}\right)$.

Traceability. This ensures that the adversary cannot produce a signature that cannot be traced to an active member of the group at the chosen epoch. In the game, the adversary can corrupt any user and even chooses the tracing key of the tracing manager. The adversary is not given the group manager's secret key as this would allow her to create dummy users which are thus untraceable. Note that unlike [LPY12b, LPY12a, NFHF09], our definition captures that a member of the group should not be able to sign w.r.t. epochs prior to her joining the group since we do not restrict the adversary's forgery to be w.r.t. to the current epoch (i.e. the current version of the group information). The adversary wins if she produces a signature whose signer cannot be identified or is an inactive member at the chosen epoch. The adversary also wins if the Judge algorithm does not accept the tracing decision on the forgery.

Formally, an $\mathcal{FDGS}$ scheme is *traceable* if for all $\lambda \in \mathbb{N}$, the advantage

$$\mathsf{Adv}^{\text{Trace}}_{\mathcal{FDGS},\mathcal{A}}(\lambda) := \Pr[\mathbf{Exp}^{\text{Trace}}_{\mathcal{FDGS},\mathcal{A}}(\lambda) = 1]$$

is negligible (in λ) for all PPT adversaries $\mathcal{A}$.

Remark 2. *To get an honestly-generated tracing key variant of the game in Fig. 7.2, we replace lines 2-5 in the game in Fig. 7.2 by the following lines:*

- $\Big((\mathsf{msk},\mathsf{mpk},\mathsf{info}),(\mathsf{tsk},\mathsf{tpk})\Big) \leftarrow \langle\!\langle \mathsf{GKGen}_{\mathcal{GM}}(\mathsf{crs}), \mathsf{GKGen}_{\mathcal{TM}}(\mathsf{crs})\rangle\!\rangle.$
- Set $\mathsf{gpk} := (\mathsf{crs},\mathsf{mpk},\mathsf{tpk})$.
- $\Big(m,\Sigma,\tau\Big) \leftarrow \mathcal{A}^{\mathsf{UToM},\mathsf{CI},\mathsf{SndToM},\mathsf{Reveal},\mathsf{Sign},\mathsf{ModifyReg},\mathsf{UpdateGroup}}\Big(\mathsf{play} : \mathsf{st}_{\mathsf{init}},\mathsf{gpk},\mathsf{info},\mathsf{tsk}\Big).$

Tracing Soundness. As recently defined by [SSE$^+$12] in the context of partially dynamic group signatures, this requirement ensures that even if both the group and the tracing managers as well as all members of the group collude, they cannot produce a valid signature that traces to two different members (this is a different, stronger setting than in the Traceability property above). Such a requirement is vital for many applications. For example, applications where signers get rewarded or where we need to stop abusers shifting blame to others.

In the definition, the adversary can fully corrupt all parties involved and wins if she produces a valid signature and valid tracing proofs that the signature traces to different (possibly corrupt) users. We may also consider a stronger variant where the adversary wins by producing a signature that traces to different epochs.

Formally, an $\mathcal{FDGS}$ scheme has *tracing soundness* if for all $\lambda \in \mathbb{N}$,

$$\mathsf{Adv}^{\text{Trace-Sound}}_{\mathcal{FDGS},\mathcal{A}}(\lambda) := \Pr[\mathbf{Exp}^{\text{Trace-Sound}}_{\mathcal{FDGS},\mathcal{A}}(\lambda) = 1]$$

is negligible (in λ) for all PPT adversaries $\mathcal{A}$.

Remark 3. *To get an honestly-generated tracing key variant of the game in Fig. 7.2, we replace lines 2-4 in the game in Fig. 7.2 by the following lines:*

- $\left(\mathsf{st}_{\mathsf{init}}, \mathsf{msk}, \mathsf{mpk}, \mathsf{info}\right) \leftarrow \mathcal{A}^{\langle \cdot, \mathsf{GKGen}_{\mathcal{TM}}(\mathsf{crs})\rangle}(\mathsf{init} : \mathsf{crs})$.
- Return 0 if $\mathsf{GKGen}_{\mathcal{TM}}$ did not accept or $\mathcal{A}$'s output is not well-formed.
- Parse the output of $\mathsf{GKGen}_{\mathcal{TM}}$ as $(\mathsf{tsk}, \mathsf{tpk})$ and set $\mathsf{gpk} := (\mathsf{crs}, \mathsf{mpk}, \mathsf{tpk})$.
- $\left(m, \Sigma, \{\mathsf{i}_i, \pi_{\mathrm{Trace}_i}\}_{i=1}^{2}, \mathsf{info}_\tau\right) \leftarrow \mathcal{A}^{\mathsf{CI}, \mathsf{ModifyReg}}\left(\mathsf{play} : \mathsf{st}_{\mathsf{init}}, \mathsf{gpk}, \mathsf{tsk}\right)$.

7.2.2 Comparison with Existing Models

Models used by accumulator-based constructions, e.g. [BS01, CL02, TX03, AST01, Ngu05, NFHF09], the vast majority of which are stated informally, are specific to that particular design paradigm and do not generalise to other construction approaches. Moreover, most of them do not take into account some of the attacks that arise in a more formal setting. For instance, some models only protect against partially but not fully corrupt tracing managers and do not capture the tracing soundness requirement. On the other hand, models used by other design approaches, e.g. [NFHF09, LPY12b, LPY12a] are also specific to those approaches and have their own shortcomings. For instance, as discussed earlier, the models used by the state-of-the-art constructions by Libert et al. [LPY12b, LPY12a] and Nakanishi et al. [NFHF09] do not prevent a group member from being able to sign w.r.t. time intervals before she joined the group. This is an attack that can be problematic in some applications of the primitive. In the traceability game used in [NFHF09] as well as the misidentification game used in [LPY12b, LPY12a], the adversary is required to output a signature that is valid w.r.t. the current interval (epoch) and therefore the definitions do not capture the attack we highlight. We stress that the authors of the concerned models never claimed that their models cover such an attack as it might not be a problem for their intended applications.

The traceability issue we shed light on does not apply to accumulator based models. In these settings, when the group changes, an update is published containing a list of the currently active group members and most constructions work by having the signer prove membership in such a list. Therefore, even if a malicious member tries to sign w.r.t. an earlier version of the group information, she still has to prove she is a member of the group at the concerned interval.

```
– $\left(m, \Sigma, \mathsf{info}_\tau\right) \leftarrow \mathcal{A}^{\mathsf{CI},\mathsf{SndToU},\mathsf{Reveal},\mathsf{Sign},\mathsf{ModifyReg}}\left(\mathsf{play} : \mathsf{st}_{\mathsf{init}}, \mathsf{gpk}\right)$.
– If $\mathsf{Verify}(\mathsf{gpk}, \mathsf{info}_\tau, m, \Sigma) = 0$ Then Return 0.
– $(\mathsf{i}, \pi_{\mathsf{Trace}}) \leftarrow \mathsf{Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_\tau, \mathsf{reg}, m, \sigma)$
– If $\mathsf{i} \notin \mathsf{HUL} \setminus \mathsf{BUL}$ or $(\mathsf{i}, m, \Sigma, \tau) \in \mathsf{SL}$ Then Return 0 Else Return 1.
```

Figure 7.3: Modified non-frameability game.

In addition [NFHF09, LPY12b, LPY12a] only consider a partially but not fully corrupt tracing manager in the non-frameability game. Moreover, they do not capture the requirement that a signature should only trace to one member (i.e. tracing soundness). The latter is vital for many applications of the primitive.

Another distinction from existing models is that our model allows maliciously generated authorities' keys when applicable. Therefore, it offers more stringent security than existing models which rely on such keys being generated honestly.

7.3 Recovering Other Models

We give security reductions which relate our model to other well-known models for group signatures. All these models assume honest key generation, for both group and tracing managers, which is a special case of our model. We consider three models. First, the model for static group signatures given in [BMW03]. We then consider two models for partially dynamic groups from [BSZ05] and [KY06].

7.3.1 Static Group Signatures [BMW03]

We note that we can recover static group signatures [BMW03] from our group signatures. We fix the group manager as the designated opener and include tsk in the group master secret key. In the setup, group members generate their key pairs and interact with the group manager to join the group. Their Open algorithm does not output proofs, as their model does not use a Judge algorithm, so we define a variant of our non-frameability game from Fig. 7.2 where we replace the last 4 lines in the game in Fig. 7.2 by the ones in Fig. 7.3.

This gives a sensible and compatible definition which allows us to recover the model from the fully dynamic scheme.

Static group signatures are just fully dynamic group signatures with no joining, issuing, or group updates. Correctness follows trivially from the correctness of the fully dynamic group signature scheme. [BMW03]-full-anonymity follows from (full) anonymity of the fully dynamic group signature scheme, while [BMW03]-full-traceability follows from our traceability and non-frameability requirements. We now give an explicit construction and security reductions which show how our model relates to [BMW03].

```
$\underline{\mathsf{GS}_{GKg}(1^\lambda, 1^n)} \rightarrow (\mathsf{GS_{gpk}}, \mathsf{GS_{gmsk}}, \mathsf{GS}_{\mathbf{gsk}})$
. $\mathsf{crs} \leftarrow \mathsf{GSetup}(1^\lambda)$.
. $\Big((\mathsf{msk}, \mathsf{mpk}, \mathsf{info}), (\mathsf{tsk}, \mathsf{tpk})\Big) \leftarrow \langle\langle \mathsf{GKGen}_{\mathcal{GM}}(\mathsf{crs}), \mathsf{GKGen}_{\mathcal{TM}}(\mathsf{crs}) \rangle\rangle$.
. Set $\mathsf{gpk} := (\mathsf{crs}, \mathsf{mpk}, \mathsf{tpk})$.
. For each user:
○ Run $(\mathbf{usk}[\mathsf{i}], \mathbf{upk}[\mathsf{i}]) \leftarrow \mathsf{UKGen}(1^\lambda)$.
○ Run $\langle \mathsf{Join}(\mathsf{info}, \mathsf{gpk}, \mathsf{i}, \mathbf{usk}[\mathsf{i}]), \mathsf{Issue}(\mathsf{info}, \mathsf{msk}, \mathsf{i}, \mathbf{upk}[\mathsf{i}]) \rangle$.
. Set $\mathsf{GS_{gmsk}} := (\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}, \mathsf{reg})$, $\mathsf{GS_{gpk}} := (\mathsf{gpk}, \mathsf{info})$
. Set $\mathsf{GS}_{\mathbf{gsk}[\mathsf{i}]} := (\mathsf{gpk}, \mathbf{gsk}[\mathsf{i}], \mathsf{info})$.

$\underline{\mathsf{GS}_{GSig}(\mathsf{GS}_{\mathbf{gsk}[\mathsf{i}]}, m)} \rightarrow \sigma$
. Parse $\mathsf{GS_{gpk}}$ as $(\mathsf{gpk}, \mathsf{info})$ and $\mathsf{GS}_{\mathbf{gsk}[\mathsf{i}]}$ as $(\mathsf{gpk}, \mathbf{gsk}[\mathsf{i}], \mathsf{info})$.
. Return $\mathsf{Sign}(\mathsf{gpk}, \mathbf{gsk}[\mathsf{i}], \mathsf{info}, m)$.

$\underline{\mathsf{GS}_{GVf}(\mathsf{GS_{gpk}}, m, \sigma)} \rightarrow 0/1$
. Parse $\mathsf{GS_{gpk}}$ as $(\mathsf{gpk}, \mathsf{info})$.
. Return $\mathsf{Verify}(\mathsf{gpk}, \mathsf{info}, m, \sigma)$.

$\underline{\mathsf{GS}_{Open}(\mathsf{GS_{gmsk}}, m, \sigma)} \rightarrow \{\mathsf{i}\} \cup \{\perp\}$
. Parse $\mathsf{GS_{gpk}}$ as $(\mathsf{gpk}, \mathsf{info})$ and $\mathsf{GS_{gmsk}}$ as $(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}, \mathsf{reg})$.
. Run $(\mathsf{i}, \pi_{\mathrm{Trace}}) \leftarrow \mathsf{Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_\tau, \mathsf{reg}, m, \sigma)$.
. Return $\mathsf{i}$.
```

Figure 7.4: Static group signatures [BMW03] from our group signatures

Theorem 10. *The construction of Fig. 7.4 is a secure static group signature scheme in the sense of [BMW03], if* GS *is correct, anonymous, non-frameable (in the Fig. 7.3 variant), traceable and tracing-sound.*

Proof. Correctness is straightforward to verify. Next we prove full-anonymity. Suppose that there exists an efficient adversary $\mathcal{B}$ who successfully breaks the [BMW03]-full-anonymity of the static group signature scheme with probability that is not negligible, with respect to a particular group of users. We construct an efficient adversary $\mathcal{A}$ for the (full) anonymity of the fully dynamic group signature scheme. While initializing the fully dynamic scheme, $\mathcal{A}$ behaves honestly, so that all parameters are honestly generated. Next, $\mathcal{A}$ adds users to the scheme using the UToM oracle to create the group of users for $\mathcal{B}$, and learns their signing keys using the Reveal oracle. Now, $\mathcal{A}$ is able to start $\mathcal{B}$ on input $(\mathsf{GS_{gpk}}, \mathsf{GS}_{\mathbf{gsk}})$. $\mathcal{A}$ can answer $\mathcal{B}$'s Open queries using her Trace oracle. When $\mathcal{B}$ outputs his challenge (i_0, i_1, m),

$\mathcal{A}$ calls $\mathsf{Chal}_b(\mathsf{info}, i_0, i_1, m)$ and gets a challenge signature σ^* which she forwards to $\mathcal{B}$ as the challenge signature. Again, using the Trace oracle, $\mathcal{A}$ can answer $\mathcal{B}$'s Open queries as long as they do not involve the challenge signature σ^*. Eventually, when $\mathcal{B}$ outputs his guess b^*, $\mathcal{A}$ returns b^* in her game. Clearly, if $\mathcal{B}$ wins his game, $\mathcal{A}$ also wins her game with the same probability. Therefore, the (full) anonymity of the full dynamic scheme implies full-anonymity of the static scheme.

Finally, we turn our attention to full-traceability. Suppose that there exists an efficient adversary $\mathcal{B}$ who successfully breaks the [BMW03]-full-traceability of the static group signature scheme w.r.t. a particular group of users with non-negligible probability. Using $\mathcal{B}$ which produces with non-negligible probability (m, σ) which cannot be opened in the [BMW03]-full-traceability game, we construct an efficient adversary $\mathcal{A}_1$ against the traceability of the fully dynamic group signature scheme. Adversary $\mathcal{A}_1$ behaves honestly while initializing the fully dynamic scheme, so that all parameters are honestly generated. As part of the initialization, $\mathcal{A}_1$ generates the tracing key tsk for the fully dynamic scheme, so she knows $\mathsf{GS}_{\mathsf{gmsk}}$. Next, $\mathcal{A}_1$ adds users to the scheme using the UToM oracle to create the group of users for $\mathcal{B}$. Now, $\mathcal{A}_1$ starts $\mathcal{B}$ on input $(\mathsf{GS}_{\mathsf{gmsk}}, \mathsf{GS}_{\mathsf{gpk}})$. Adversary $\mathcal{A}_1$ answers signature queries from $\mathcal{B}$ using her own Sign oracle if the user requested by $\mathcal{B}$ has not been already corrupted. If the secret key of the user in question has already been revealed to $\mathcal{B}$, $\mathcal{A}_1$ uses the user's secret signing key to answer signing queries w.r.t. that user. $\mathcal{A}_1$ is also able to answer $\mathcal{B}$'s corrupt queries using her Reveal oracle. When $\mathcal{B}$ outputs (m, σ), the probability that σ cannot be opened is non-negligible, so $\mathcal{A}_1$ can use these to output (m, σ, τ) to break the traceability of the fully dynamic scheme.

Therefore, if traceability holds, we conclude that when $\mathcal{B}$ produces (m, σ) and successfully breaks full-traceability, we have with overwhelming probability that $\mathsf{Open}(\mathsf{gmsk}, m, \sigma) = i$ for some i. This will allow us to construct an efficient adversary $\mathcal{A}_2$ against the non-frameability of the fully dynamic group signature scheme. Similarly to $\mathcal{A}_1$, adversary $\mathcal{A}_2$ behaves honestly while initializing the fully dynamic scheme, so that all parameters are honestly generated. As part of the initialization, $\mathcal{A}_2$ gets the tracing manager's key tsk for the fully dynamic scheme, so she knows $\mathsf{GS}_{\mathsf{gmsk}}$. Note that unlike $\mathcal{A}_1$, $\mathcal{A}_2$ additionally has the full dynamic group signature group manager's secret key msk. $\mathcal{A}_2$ can add users to the group using her SndToU oracle. Now, $\mathcal{A}_2$ starts $\mathcal{B}$ on input $(\mathsf{GS}_{\mathsf{gmsk}}, \mathsf{GS}_{\mathsf{gpk}})$. $\mathcal{A}_2$ answers $\mathcal{B}$'s sign queries using her own Sign oracle if the user in question is honest or directly if she knows the secret key of the user. Also, she can reveal the user's secret key requested by $\mathcal{B}$ using her Reveal oracle. Now, when $\mathcal{B}$ breaks full-traceability, he outputs (m, σ) such that $\mathsf{Open}(\mathsf{gmsk}, m, \sigma) = i$ for some i, with overwhelming probability. It follows that (m, σ) is a valid signature, and that $\mathcal{B}$ did not request a signature for m from user i, nor did $\mathcal{B}$ request the signing key of i. Therefore, $\mathcal{A}_2$ can use (m, σ) to break the variant of non-frameability (Fig. 7.3) of the fully dynamic group signature scheme, with probability negligibly different from the success probability of $\mathcal{B}$. This shows that full-traceability of the static scheme of [BMW03] is implied by the traceability and non-frameability of the fully dynamic scheme. □

7.3.2 Partially Dynamic Group Signatures [BSZ05]

Fully dynamic group signatures also imply the partially dynamic group signatures of [BSZ05] in the case where nobody is removed from the group. Anonymity, non-frameability and traceability all follow from our corresponding definitions. Correctness follows trivially from the correctness of the fully dynamic group signature scheme.

We now give an explicit construction (in Fig. 7.5) and security reductions which show how our model relates to [BSZ05].

Theorem 11. *The construction in Fig. 7.5 is a secure partially dynamic group signature scheme in the sense of [BSZ05], if* GS *is correct, anonymous, non-frameable, traceable and tracing-sound.*

Proof. Correctness follows trivially from the correctness of the fully dynamic group signature scheme.

Next we prove anonymity. Suppose that there exists an efficient adversary $\mathcal{B}$ who successfully breaks the [BSZ05]-anonymity of the partially dynamic group signature scheme. We construct an efficient adversary $\mathcal{A}$ for the anonymity of the fully dynamic group signature scheme. While initializing the fully dynamic scheme, $\mathcal{A}$ behaves honestly, so that all parameters are honestly generated, but $\mathcal{A}$ knows the group manager's secret key msk, and thus knows $\mathsf{DGS_{ik}}$. Adversary $\mathcal{A}$ sets $\mathsf{DGS_{gpk}} := \mathsf{gpk} = (\mathsf{crs}, \mathsf{mpk}, \mathsf{tpk}, \mathsf{info})$ and starts $\mathcal{B}$ on $(\mathsf{DGS_{gpk}}, \mathsf{DGS_{ik}})$. Adversary $\mathcal{B}$ has access to oracles $\mathsf{Ch}, \mathsf{Open}, \mathsf{SndToU}, \mathsf{WReg}, \mathsf{USK}, \mathsf{CrptU}$, which directly correspond to the oracles that $\mathcal{A}$ has access to, namely $\mathsf{Chal}_b, \mathsf{Trace}, \mathsf{SndToU}$, $\mathsf{ModifyReg}, \mathsf{Reveal}, \mathsf{CrptU}$. Therefore, $\mathcal{A}$ can simulate all necessary oracles for $\mathcal{B}$. When $\mathcal{B}$ calls the challenge oracle Ch on (i_0, i_1, m), adversary $\mathcal{A}$ calls her challenge oracle Chal_b on $(\mathsf{info}_{\tau_{\text{current}}}, i_0, i_1, m)$ where τ_{current} is the current

```
$\mathsf{DGS}_{GKg}(1^\lambda) \rightarrow (\mathsf{DGS}_{\mathsf{gpk}}, \mathsf{DGS}_{\mathsf{ik}}, \mathsf{DGS}_{\mathsf{ok}})$
. $\mathsf{crs} \leftarrow \mathrm{GSetup}(1^\lambda)$.
. $\Big((\mathsf{msk}, \mathsf{mpk}, \mathsf{info}), (\mathsf{tsk}, \mathsf{tpk})\Big) \leftarrow \langle\!\langle \mathsf{GKGen}_{\mathcal{GM}}(\mathsf{crs}), \mathsf{GKGen}_{\mathcal{TM}}(\mathsf{crs}) \rangle\!\rangle$.
. Set $\mathsf{gpk} := (\mathsf{crs}, \mathsf{mpk}, \mathsf{tpk}, \mathsf{info})$.
. Set $\mathsf{DGS}_{\mathsf{gpk}} := \mathsf{gpk}$, $\mathsf{DGS}_{\mathsf{ik}} := (\mathsf{gpk}, \mathsf{msk})$ and $\mathsf{DGS}_{\mathsf{ok}} := (\mathsf{gpk}, \mathsf{tsk})$.

$\mathsf{DGS}_{UKg}(1^\lambda) \rightarrow (\mathsf{DGS}_{\mathbf{upk}[i]}, \mathsf{DGS}_{\mathbf{usk}[i]})$
. Return $\mathsf{UKGen}(1^\lambda)$.

$\langle \mathrm{Join}, \mathrm{Iss} \rangle$
. Run $\langle \mathsf{Join}(\mathsf{info}_{\tau_{\mathrm{current}}}, \mathsf{gpk}, i, \mathbf{usk}[i]), \mathsf{Issue}(\mathsf{info}_{\tau_{\mathrm{current}}}, \mathsf{msk}, i, \mathbf{upk}[i]) \rangle$,
with the issuer modifying $\mathsf{reg}[i]$ if accepting, and the Join algorithm
outputting $\mathsf{DGS}_{\mathbf{gsk}}[i] = (\mathsf{gpk}, \mathbf{gsk}[i], \mathsf{info}_\tau)$.

$\mathsf{DGS}_{GSig}(\mathsf{DGS}_{\mathsf{gpk}}, \mathsf{DGS}_{\mathbf{gsk}[i]}, m) \rightarrow \sigma$
. Parse $\mathsf{DGS}_{\mathsf{gpk}}$ as $\mathsf{gpk}$ and $\mathsf{DGS}_{\mathbf{gsk}}[i]$ as $(\mathsf{gpk}, \mathbf{gsk}[i], \mathsf{info}_\tau)$.
. $\Sigma \leftarrow \mathrm{Sign}(\mathsf{gpk}, \mathbf{gsk}[i], \mathsf{info}_{\tau_{\mathrm{current}}}, m)$.
. Return $\sigma = (\Sigma, \tau_{\mathrm{current}})$.

$\mathsf{DGS}_{GVf}(\mathsf{DGS}_{\mathsf{gpk}}, m, \sigma) \rightarrow 0/1$
. Parse $\mathsf{DGS}_{\mathsf{gpk}}$ as $\mathsf{gpk}$ and $\sigma$ as $(\Sigma, \tau)$.
. Return $\mathsf{Verify}(\mathsf{gpk}, \mathsf{info}_\tau, m, \Sigma)$.

$\mathsf{DGS}_{Open}(\mathsf{DGS}_{\mathsf{gpk}}, \mathsf{DGS}_{\mathsf{ok}}, \mathsf{DGS}_{\mathsf{reg}}, m, \sigma) \rightarrow (i, \pi_{\mathrm{Trace}})$
. Set $\mathsf{DGS}_{\mathsf{gpk}}$ as $\mathsf{gpk}$, $\mathsf{DGS}_{\mathsf{ok}}$ as $(\mathsf{gpk}, \mathsf{tsk})$ and $\sigma$ as $(\Sigma, \tau)$.
. Return $\mathsf{Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_\tau, \mathsf{reg}, m, \Sigma)$.

$\mathsf{DGS}_{Judge}(\mathsf{DGS}_{\mathsf{gpk}}, j, \mathsf{DGS}_{\mathbf{upk}[j]}, m, \sigma, \pi_{\mathrm{Trace}}) \rightarrow 0/1$
. Parse $\mathsf{DGS}_{\mathsf{gpk}}$ as $\mathsf{gpk}$ and $\sigma$ as $(\Sigma, \tau)$.
. Return $\mathsf{Judge}(\mathsf{gpk}, j, \mathsf{info}_\tau, \pi_{\mathrm{Trace}}, \mathbf{upk}[j], m, \Sigma)$.
```

Figure 7.5: Group signatures [BSZ05] from our group signatures.

epoch. Once $\mathcal{A}$ gets back a signature Σ from her oracle, she passes $\sigma = (\Sigma, \tau_{\mathrm{current}})$ to $\mathcal{B}$ as the answer. Whenever $\mathcal{B}$ outputs his final bit guess b^*, $\mathcal{A}$ returns b^* as her answer. We have that $\mathcal{A}$ breaks anonymity of the fully dynamic group signature whenever $\mathcal{B}$ successfully breaks anonymity of the partially dynamic scheme. Thus, the former anonymity definition implies the latter.

Next, we prove traceability. Assume there exists an efficient adversary $\mathcal{B}$ who successfully breaks the [BSZ05]-traceability of the partially dynamic scheme. We construct an efficient adversary $\mathcal{A}$ against the traceability of the fully dynamic group signature scheme. While initializing the fully dynamic scheme, $\mathcal{A}$ behaves honestly, so that all parameters are honestly generated, but $\mathcal{A}$ knows the tracing manager's secret key tsk, so she knows $\mathsf{DGS_{ok}}$. Adversary $\mathcal{A}$ sets $\mathsf{DGS_{gpk}} := \mathsf{gpk} = (\mathsf{crs}, \mathsf{mpk}, \mathsf{tpk}, \mathsf{info})$ and starts $\mathcal{B}$ on $(\mathsf{DGS_{gpk}}, \mathsf{DGS_{ok}})$. Adversary $\mathcal{B}$ has access to oracles $\mathsf{SndToI}, \mathsf{AddU}, \mathsf{RReg}, \mathsf{USK}, \mathsf{CrptU}$, which directly correspond to some of the oracles $\mathcal{A}$ has access to, namely $\mathsf{SndToM}, \mathsf{AddU}, \mathsf{ReadReg}, \mathsf{Reveal}, \mathsf{CrptU}$. Therefore, $\mathcal{A}$ can simulate all necessary oracles for $\mathcal{B}$. When $\mathcal{B}$ outputs (m, σ) where $\sigma = (\Sigma, \tau)$, $\mathcal{A}$ returns (m, Σ, τ) in her game. Now, whenever $\mathcal{B}$ breaks [BSZ05]-traceability, $\mathcal{A}$ breaks the traceability of the fully dynamic scheme. This shows that traceability of fully dynamic schemes implies the same for partially dynamic schemes, in the sense of [BSZ05].

Finally, we prove non-frameability. Suppose that there exists an efficient adversary $\mathcal{B}$ who successfully breaks the [BSZ05]-non-frameability of the partially dynamic group signature scheme. We construct an efficient adversary $\mathcal{A}$ against the non-frameability of the fully dynamic group signature scheme. While initializing the fully dynamic scheme, $\mathcal{A}$ behaves honestly, so that all parameters are honestly generated, but $\mathcal{A}$ knows the tracing manager's secret key tsk and the issuer's secret key msk. Therefore, $\mathcal{A}$ knows $\mathsf{DGS_{ik}}, \mathsf{DGS_{ok}}$. Adversary $\mathcal{A}$ sets $\mathsf{DGS_{gpk}} := \mathsf{gpk} = (\mathsf{crs}, \mathsf{mpk}, \mathsf{tpk}, \mathsf{info})$ and starts $\mathcal{B}$ on $(\mathsf{DGS_{gpk}}, \mathsf{DGS_{ok}}, \mathsf{DGS_{ik}})$. Adversary $\mathcal{B}$ has access to oracles $\mathsf{SndToU}, \mathsf{WReg}, \mathsf{GSig}, \mathsf{USK}, \mathsf{CrptU}$, which directly correspond to some of the oracles that $\mathcal{A}$ has access to, namely $\mathsf{SndToU}, \mathsf{ModifyReg}, \mathsf{Sign}, \mathsf{Reveal}, \mathsf{CrptU}$. Therefore, $\mathcal{A}$ can simulate all necessary oracles for $\mathcal{B}$. Whenever $\mathcal{B}$ outputs $(m, \sigma, i, \pi_{\mathrm{Trace}})$ where $\sigma = (\Sigma, \tau)$, $\mathcal{A}$ returns $(m, \Sigma, i, \pi_{\mathrm{Trace}}, \mathsf{info}_\tau)$ in her game. Therefore, whenever $\mathcal{B}$ breaks the non-frameability of the partially dynamic scheme, $\mathcal{A}$ succeeds in breaking the non-frameability of the fully dynamic scheme. This shows that non-frameability of fully dynamic schemes implies the same for partially dynamic schemes, in the sense of [BSZ05]. □

7.3.3 Partially Dynamic Group Signatures [KY06]

Finally, we consider the partially-dynamic model of [KY06]. We fix the group manager as the designated opener and set $(\mathsf{msk}, \mathsf{tsk})$ to be the group master secret key. Our group info and registration table generalise their public state string. Their Join algorithm runs our user key-generation and $\mathsf{Join/Issue}$ algorithms. The membership certificate is then the user's public key along with the group information, and the membership secret is the user's private key. Again, their Open algorithm does not output proofs, and the model does not have a judge algorithm. Therefore, as in the case of [BMW03] we modify our non-frameability game from Fig. 7.2 where we replace the last 4 lines in the game in Fig. 7.2 with those in Fig. 7.3.

Correctness follows trivially from the correctness of the fully dynamic group signature scheme. Security against misidentification-attacks follows from traceability, security against framing-attacks follows from non-frameability, and anonymity follows from the (full) anonymity of the fully dynamic group signature.

We now give an explicit construction (in Fig. 7.6) and security reductions which show how our model relates to [KY06].

```
Setup(1^λ) → (Y, S)
. $\mathsf{crs} \leftarrow \mathsf{GSetup}(1^\lambda)$.
. $\big((\mathsf{msk},\mathsf{mpk},\mathsf{info}),(\mathsf{tsk},\mathsf{tpk})\big) \leftarrow \langle \mathsf{GKGen}_{\mathcal{GM}}(\mathsf{crs}), \mathsf{GKGen}_{\mathcal{TM}}(\mathsf{crs})\rangle$.
. Set $\mathsf{gpk} := (\mathsf{crs},\mathsf{mpk},\mathsf{tpk})$.
. Set $\mathcal{S} := (\mathsf{msk},\mathsf{tsk})$, $\mathcal{Y} := (\mathsf{gpk},\mathsf{info})$.
. Set $\mathsf{St} := (\mathsf{St}_{users},\mathsf{St}_{trans}) := (\emptyset,\mathsf{reg})$.

Join
This is an interactive protocol between a user and the group manager.
. The user runs $(\mathbf{usk}[\mathrm{i}],\mathbf{upk}[\mathrm{i}]) \leftarrow \mathsf{UKGen}(1^\lambda)$.
. Both parties run $\langle \mathsf{Join}(\mathsf{info}_{\tau_{\mathrm{current}}},\mathsf{gpk},\mathrm{i},\mathbf{usk}[\mathrm{i}]), \mathsf{Issue}(\mathsf{info}_{\tau_{\mathrm{current}}},\mathsf{msk},\mathrm{i},\mathbf{upk}[\mathrm{i}])\rangle$.
. The user receives private output $\lfloor i,\mathsf{cert}_i,\mathsf{sec}_i \rfloor := \lfloor \mathrm{i},(\mathbf{upk}[\mathrm{i}],\mathsf{info}_\tau),\mathbf{gsk}[\mathrm{i}] \rfloor$
. Set $\mathsf{transcript}_i := \mathsf{reg}[\mathrm{i}]$.
. After a successful execution, the state is updated,
with $\mathsf{St}_{users} := \mathsf{St}_{users} \cup \{i\}$ and $\mathsf{St}_{trans} := \mathsf{St}_{trans} || \lfloor i,\mathsf{transcript}_i \rfloor$.

Sign(Y, cert_i, sec_i, m) → σ
. Parse $\mathcal{Y}$ as $(\mathsf{gpk},\mathsf{info})$ and $\mathsf{sec}_i$ as $\mathbf{gsk}[\mathrm{i}]$.
. $\Sigma \leftarrow \mathsf{Sign}(\mathsf{gpk},\mathbf{gsk}[\mathrm{i}],\mathsf{info}_{\tau_{\mathrm{current}}},m)$.
. Return $\sigma = (\Sigma,\tau_{\mathrm{current}})$.

Verify(Y, m, σ) → ⊤/⊥
. Parse $\mathcal{Y}$ as $(\mathsf{gpk},\mathsf{info})$ and $\sigma$ as $(\Sigma,\tau)$.
. Let $b = \mathsf{Verify}(\mathsf{gpk},\mathsf{info}_\tau,m,\Sigma)$.
. Return $\top$ if $b = 1$, or $\bot$ if $b = 0$.

Open(m, σ, Y, S, St) → i ∈ St_users ∪ {⊥}
. Parse $\mathcal{Y}$ as $(\mathsf{gpk},\mathsf{info})$, $\mathcal{S}$ as $(\mathsf{msk},\mathsf{tsk})$, $\sigma$ as $(\Sigma,\tau)$, and $\mathsf{St}_{trans}$ as reg.
. Run $(i,\pi_{\mathrm{Trace}}) \leftarrow \mathsf{Trace}(\mathsf{gpk},\mathsf{tsk},\mathsf{info}_\tau,\mathsf{reg},m,\Sigma)$.
. Return $i$ if $i \in \mathsf{St}_{users}$, or $\bot$ if $i = 0$.
```

Figure 7.6: Group signatures [KY06] from our group signatures

Theorem 12. *The construction in Fig. 7.6 is is a secure partially dynamic group signature scheme in the sense of [KY06] if* GS *is correct, anonymous, non-frameable, traceable and tracing-sound. .*

Proof. We begin by proving correctness. User tagging soundness and join soundness are both trivial by construction of the `Setup` and `Join` algorithms. Signing soundness and opening soundness both follow trivially from the correctness of the fully dynamic group signature scheme.

Next we prove security against misidentification-attacks. Suppose that $\mathcal{B}$ is an efficient adversary against the misidentification-attack game of [KY06]. As stated in [KY06], without loss of generality, we may consider an adversary who controls all users in the system, and wins the misidentification-attack game by providing a signature which fails to open to any user. We construct an efficient adversary $\mathcal{A}$ against the traceability of the fully dynamic group signature scheme. While initializing the fully dynamic scheme, $\mathcal{A}$ behaves honestly, so that all parameters are honestly generated, but $\mathcal{A}$ knows the tracing key tsk. In the misidentification-attack game, $\mathcal{B}$ has access to oracles $\mathcal{Q}_{pub}, \mathcal{Q}_{read}, \mathcal{Q}_{open}$ and $\mathcal{Q}_{a-join}$. Adversary $\mathcal{A}$ can simulate all of those oracles using the information available to her in her traceability game as well as the oracles she has access to in her game. The oracle $\mathcal{Q}_{pub}$ is easy to simulate. The oracle $\mathcal{Q}_{read}$ corresponds directly to $\mathsf{ReadReg}$. The oracle $\mathcal{Q}_{open}$ can be easily simulated by $\mathcal{A}$, since $\mathcal{A}$ possesses the tracing key tsk. Also, $\mathcal{Q}_{a-join}$ can be simulated using CrptU and SndToM. Therefore, $\mathcal{A}$ can run $\mathcal{B}$, successfully simulating all oracles, and when $\mathcal{B}$ outputs (m, σ), where $\sigma = (\Sigma, \tau)$, $\mathcal{A}$ outputs (m, Σ, τ) as her answer in her traceability game. Whenever $\mathcal{B}$ succeeds in breaking security against misidentification attacks, $\mathcal{A}$ breaks traceability. Therefore, traceability of the fully dynamic group signature implies security against misidentification attacks.

Next, we prove security against framing-attacks. Suppose that $\mathcal{B}$ is an efficient adversary against the framing-attack game of [KY06]. We construct an efficient adversary $\mathcal{A}$ against the non-frameability of the fully dynamic group signature scheme. While initializing the fully dynamic scheme, $\mathcal{A}$ behaves honestly, so that all parameters are honestly generated, but $\mathcal{A}$ knows $\mathsf{msk}, \mathsf{tsk}$. In his game, $\mathcal{B}$ has access to oracles $\mathcal{Q}_{pub}, \mathcal{Q}_{key}, \mathcal{Q}_{b-join}, \mathcal{Q}_{read}, \mathcal{Q}_{write}, \mathcal{Q}_{sign}$. Using the information available to her in her game as well as the oracles she has access to in her non-frameability game, $\mathcal{A}$ can simulate all of $\mathcal{B}$'s oracles. The oracle $\mathcal{Q}_{pub}$ is easy to simulate. The oracle $\mathcal{Q}_{key}$ is easy to simulate since $\mathcal{A}$ knows msk and tsk. The oracle $\mathcal{Q}_{b-join}$ directly corresponds to SndToU. Oracle $\mathcal{Q}_{read}, \mathcal{Q}_{write}$ are easily simulated by $\mathcal{A}$ since $\mathcal{A}$ possesses msk and has access to the $\mathsf{ModifyReg}$ oracle. Also, $\mathcal{A}$ can simulate the $\mathcal{Q}_{sign}$ oracle for $\mathcal{B}$ using her own Sign oracle. This means that $\mathcal{A}$ can run $\mathcal{B}$ to obtain (m, σ) where $\sigma = (\Sigma, \tau)$. Note in [KY06], users created by $\mathcal{Q}_{b-join}$ are honest users whose private keys have not been revealed to $\mathcal{B}$. Therefore, a user i framed by $\mathcal{B}$ will be in $\mathsf{HUL} \setminus \mathsf{BUL}$, unless $\mathcal{A}$ explicitly calls her Reveal oracle on i. Given the output (m, σ) produced by $\mathcal{B}$, $\mathcal{A}$ outputs $(m, \Sigma, \mathsf{info}_\tau)$ as her answer in her game. Adversary $\mathcal{A}$ successfully breaks the non-frameability of the fully dynmic group signatures whenever $\mathcal{B}$ wins in his framing-attack game. Therefore, non-frameability of our fully dynamic group signatures implies security against framing attacks in the sense of [KY06].

Finally, we prove anonymity. Suppose $\mathcal{B}$ is an efficient adversary against the anonymity game of [KY06]. We construct an efficient adversary $\mathcal{A}$ against the anonymity of the fully-dynamic group signature scheme. While initializing the fully dynamic scheme, $\mathcal{A}$ behaves honestly, so that all parameters are honestly generated, but $\mathcal{A}$ knows msk. In the anonymity game of [KY06], $\mathcal{B}$ has access to oracles $\mathcal{Q}_{pub}, \mathcal{Q}_{a-join}, \mathcal{Q}_{read}, \mathcal{Q}_{open}$. The oracle $\mathcal{Q}_{pub}$ is easy to simulate. The oracle $\mathcal{Q}_{a-join}$ can be easily simulated using CrptU and knowledge of msk. Similarly, $\mathcal{Q}_{read}$ can be simulated using msk. Lastly, $\mathcal{Q}_{open}$ can be simulated by $\mathcal{A}$ by making use of her own Trace oracle. This means that $\mathcal{A}$ can simulate the anonymity game for $\mathcal{B}$, using Chal_b on the current epoch τ to provide $\mathcal{B}$ with a challenge signature $\sigma = (\Sigma, \tau)$. Now, when $\mathcal{B}$ returns a bit b^*, $\mathcal{A}$ returns that as her answer in her game. It is clear that whenever $\mathcal{B}$ succeeds in winning the [KY06] anonymity game, $\mathcal{A}$ breaks (full) anonmity of the fully dynamic group signature. Therefore, anonymity of fully dynamic group signatures implies that of partially dynamic group signatures in the sense of [KY06]. □

7.4 On the Security of Some Existing Schemes

Here we take a closer look at some of the existing fully dynamic schemes and investigate whether or not they are secure using our proposed model.

We show that the state-of-the-art certificate-based schemes in [LPY12b, LPY12a] and [NFHF09] are all susceptible to an attack against traceability which allows any user to sign w.r.t. an epoch predating her joining. In our model this directly breaks traceability, as the signature is w.r.t. an epoch in which the signer was not active. We note that our attack does not contradict the original security proofs of the schemes, but instead highlights that our definition is stronger. We also show that it is easy to repair the schemes at a reasonable cost.

At first glance, our attack is the dual of a well known issue with many revocation systems. If a user is revoked and anonymity is maintained, the revoked user is able to produce back-dated signatures that still verify. The difference here is that while the revoked user *was* authorised to be part of the group for the epoch in question, in our attack the signing user was in fact *not* authorised to sign for the group. If the adversary is able to block the opening of this signature (e.g. via legal action), its existence would implicitly frame the group's past membershipas the signature would be attributed to them.

7.4.1 Libert et al. Schemes [LPY12b, LPY12a]

In [LPY12a], users are assigned leaves of a complete binary tree and given a membership certificate containing a unique tag identifying the user, and a commitment to the path from the root to the user's leaf in the tree. Note that the certificate is not bound to the epoch at which the user joined the group. In fact, users joining does not change info_τ or the epoch τ itself.

Revocation is based on the subset difference method [NNL01], using disjoint sets S_{k_i,u_i} for $i = 1, \dots, m$ which cover non-revoked users. Sets are represented by two nodes, a node k_i and one of its descendants node u_i, and cover all leaves of the sub-tree rooted at node k_i which are not leaves of the sub-tree rooted at u_i. Revocations trigger epoch changes with info_τ updated with a new cover.

To sign, the group member anonymously proves that she holds a membership certificate, and that the node indicated by the certificate belongs to one of those sets. More precisely, the user proves that her leaf is a descendant of node k_i but not a descendant of node u_i for some $i \in [m]$.

Since user certificates are not bound to epochs and leaves are covered until their corresponding users are revoked, it is simple to break traceability: a user can join and then produce a signature for an epoch that predates her joining. A similar argument also applies to the variant of the scheme given in [LPY12b].

Theorem 13. *The fully dynamic scheme of Libert et al. [LPY12a] does not satisfy our traceability definition even w.r.t. honestly generated tracing manager's keys.*

Proof. Consider the following strategy in the traceability experiment: the adversary asks to join as a user i_1 at epoch τ_1. User i_1 gets assigned the leaf l_1. Then at a later epoch, τ_2, the adversary asks to join as a second user i_2. Finally, the adversary signs using the credentials of i_2 but for epoch τ_1.

We can check by inspection that all subproofs in the back-dated signature go through. The crucial observation is that at epoch τ_1, the leaf l_2 is not revoked and thus must be covered by one of the S_{k_i,u_i} sets. As the proof verifies and i_2 used a legitimate certificate, opening the signature will be successful and indicate i_2 as the signer. The adversary wins, as i_2 was not active at epoch τ_1. □

A possible countermeasure against the above attack is to regard unassigned leaves as revoked until they are assigned. This is simple to do as the scheme does not bound the number of revoked users. We do however need to re-examine the number of subsets required to express this, as the $2|\mathcal{R}|-1$ bound for $|\mathcal{R}|$ revoked users may now seem impractical. If we assume leaves are allocated sequentially to users, we can bound the number of subsets by $2|\mathcal{R}_1|+log(|\mathcal{N}\setminus\mathcal{R}_2|)$ where $\mathcal{R}_2$ is the set of leaves pending allocation and $\mathcal{R}_1$ is the set of leaves allocated to users who were later revoked. Thus, our fix is only marginally more expensive than the base system and much more efficient than a naive analysis would indicate.

If proving set membership/intervals can be done efficiently (and depending on how the epoch counter is implemented), another possible fix is to bind membership certificates to the join epoch and then get the signer to prove that their join epoch is not later than the signing epoch.

7.4.2 Nakanishi et al. Scheme [NFHF09]

The scheme of Nakanishi et al. [NFHF09] is another certificate-based scheme in the random oracle model. It achieves constant time for both signing and signature verification, relative to the size of the group and the number of revoked users.

A user's group membership certificate consists of a signature on (x, ID) produced by the group manager, where x is a secret owned by the user and ID is a unique integer the manager assigned to her. The group manager can revoke users by issuing revocation lists info_τ. Each list consists of a sequence of *open* integer intervals (R_i, R_{i+1}) signed by the manager, whose endpoints are all the revoked ID's. At each epoch τ, a signer fetches the current info_τ and proves, as part of the signature, that her ID is contained in one interval of the revocation list. If the ID lies between two revoked users' identities, it means it is not an endpoint and so she has not been revoked.

As in other certificate-based constructions, verifiers only know of revoked members, not active ones and, similarly to [LPY12a], the time of joining is not taken into account. This allows users to sign with respect to any epoch prior to joining the group, which represents an attack against our traceability definition.

Theorem 14. *The Nakanishi et al. [NFHF09] fully dynamic group signature scheme does not satisfy our traceability definition.*

Proof. Let $\mathcal{A}$ be an adversary against the traceability game. The adversary adds user i to the group at epoch τ. Since the user is not revoked, her ID is not an endpoint in any interval of the revocation list info_{τ}, as for all previous epochs. Therefore, $\mathcal{A}$ could easily produce valid signatures for i to any epoch $\bar{\tau} < \tau$. Since these signatures trace back to a user which was inactive at the interval with which the signature is associated, $\mathcal{A}$ succeeds in the traceability game. $\square$

The scheme could be easily immunised against the above attack. A first solution, as for [LPY12a], is to initialise the revocation list with all ID's of users that have not joined the group yet. When the manager assigns an ID to a new user, he updates reg and the revocation list info_{τ}. This way, the signature size is not affected. On the other hand, revocation lists are now proportional to the size of the maximum number of users, instead of the number of revoked users.

An alternative countermeasure requires the group manager to include the joining epochs in the certificates by signing $(x, \mathrm{ID}, \tau_{\mathrm{join}})$, where x is a secret owned by user ID and τ_{join} is the joining epoch. A signer then needs to include in the signature a proof that τ_{join} is not greater than the signing epoch. To realise the latter, one can use membership proof techniques from [TS06, CCS08] which are already used in the original scheme. This would increase the cost of signing and verifying by only a constant factor. The new membership proof would require the group manager to provide signatures for every elapsed epoch, which could be appended, for instance, to the revocation list. This makes revocation lists grow linearly with the number of revoked users as well as the number of epochs.

7.4.3 Scheme based on Sect. 6.3

In Sect. 6.3, we gave a generic construction of accountable ring signatures, where every signature can be traced back to a user in the ring. We also showed how one can obtain fully dynamic group signatures from accountable ring signatures and gave an efficient instantiation in the random oracle model, based on the DDH assumption. That instantiation yields signatures of logarithmic size (w.r.t. the size of the ring), while signing is quasi-linear, and signature verification requires a linear number of operations.

Each user has a secret key and an associated verification key. To sign, users first encrypt their verification key. Then, via a membership proof, they provide a signature of knowledge showing that the verification key belongs to the ring, and that they know the corresponding secret key. We will now prove this construction is secure w.r.t. the stronger variant of our model i.e. w.r.t. adversarially generated authorities' keys..

In what follows, note that each epoch τ specifies an instance of the accountable ring signature scheme with ring R_τ. Algorithms from the accountable ring signature scheme are labelled with ARS. We assume that the epoch can be appended to a message using the $||$ operation, and removed again without ambiguity.

An accountable ring signature scheme does not involve a group manager. Hence, to construct a group signature, we assume the existence of a functionality GMg, which allows a group manager's key-pair to be derived as $(\mathsf{gmk}, \mathsf{msk}) \leftarrow \mathrm{GMg}(1^\lambda)$, and initialises a group information board which is visible to all parties, but can only be modified by a party with msk.

The construction of a fully dynamic group signature from an accountable ring signature is presented in Fig. 7.7.

Theorem 15. *The generic group signature scheme construction (Sect. 6.2.1) from the accountable ring signature scheme of Sect. 6.3 satisfies our definitions for a secure, fully-dynamic group signature scheme.*

Proof. We begin by proving correctness. For simplicity and without loss in generality, we reduce the computational variant of the correctness requirement, i.e. where the last three lines in the correctness game in Fig. 7.2 are dropped (see Section 7.2.1), to the perfect correctness of the accountable ring signature. Let $\mathcal{A}$ be an adversary against the (computational) correctness of the fully dynamic group signature scheme. We construct an adversary $\mathcal{B}$ against the (perfect) correctness of the accountable ring signature scheme. On receiving (pp, sk) from his game, $\mathcal{B}$ sets $\mathsf{crs} = pp$ and chooses $(\mathsf{tpk}, \mathsf{tsk})$ and $(\mathsf{mpk}, \mathsf{msk})$ by himself. He also initialises reg and info. Note that $\mathcal{B}$ can compute vk corresponding to sk by computing $vk = \mathsf{ARS}_{UKGen}(pp, sk)$. Now, $\mathcal{B}$ sets $\mathsf{gpk} := (\mathsf{crs}, \mathsf{mpk}, \mathsf{tpk})$ and starts $\mathcal{A}$ on $(\mathsf{gpk}, \mathsf{info})$. Since $\mathcal{B}$ knows msk, he can simulate all oracle queries for $\mathcal{A}$. Let q be a polynomial upper

$\underline{\mathsf{GSetup}(1^\lambda)} \to \mathsf{crs}$
- Compute $\mathsf{crs} := pp \leftarrow \mathsf{ARS}_{Setup}(1^\lambda)$.
- Initialise reg and a counter $\tau := 0$.

$\underline{\langle \mathsf{GKGen}_{\mathcal{GM}}(\mathsf{crs}), \mathsf{GKGen}_{\mathcal{TM}}(\mathsf{crs})\rangle}$
- The group and tracing managers participate in an interactive protocol to generate $(\mathsf{tpk}, \mathsf{tsk}) \leftarrow \mathsf{ARS}_{OKGen}(pp)$ and $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{GMg}(1^\lambda)$.
- Set $\mathsf{gpk} := (\mathsf{crs}, \mathsf{mpk}, \mathsf{tpk})$, and initialise info.

$\underline{\mathsf{UKGen}(1^\lambda) \to (\mathbf{usk}[\mathsf{i}], \mathbf{upk}[\mathsf{i}])}$
- Return $(\mathbf{usk}[\mathsf{i}], \mathbf{upk}[\mathsf{i}]) \leftarrow \mathsf{ARS}_{UKGen}(pp)$.

$\underline{\langle \mathsf{Join}(\tau, \mathsf{gpk}, \mathsf{i}, \mathbf{usk}[\mathsf{i}]), \mathsf{Issue}(\tau, \mathsf{msk}, \mathsf{i}, \mathbf{upk}[\mathsf{i}])\rangle}$
- The group manager and a new user i who already has a key-pair undergo an interactive protocol to register i.
- Upon completion, the group manager uses msk to update reg with the transcript.
- Set $\mathbf{gsk}[\mathsf{i}] = \mathbf{usk}[\mathsf{i}]$.

$\underline{\mathsf{UpdateGroup}(\mathsf{gpk}, \mathsf{msk}, \mathsf{info}_{\text{current}}, \mathcal{S}, \mathsf{reg})} \to \mathsf{info}_{\text{new}}$
- If R_{new} is different from R_{current} then increment the counter τ
- The new group information $\mathsf{info}_{\text{new}}$ contains the current group R_{new} and the public keys of the members as well as the counter τ.

$\underline{\mathsf{Sign}(\mathsf{gpk}, \mathbf{gsk}[\mathsf{i}], \mathsf{info}_\tau, m)} \to \Sigma$
- Denote the group for the current epoch by R_τ. This is part of info_τ.
- Return $\Sigma \leftarrow \mathsf{ARS}_{Sign}(\mathsf{tpk}, m||\tau, R_\tau, \mathbf{gsk}[\mathsf{i}])$.

$\underline{\mathsf{Verify}(\mathsf{gpk}, \mathsf{info}_\tau, m, \Sigma)} \to 1/0$
- Return $\mathsf{ARS}_{Vfy}(\mathsf{tpk}, m||\tau, R_\tau, \Sigma)$.

$\underline{\mathsf{Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_\tau, \mathsf{reg}, m, \Sigma)} \to (\mathsf{i}, \pi_{\text{Trace}})$
- Return $\mathsf{ARS}_{Open}(m||\tau, R_\tau, \Sigma, \mathsf{tsk})$.

$\underline{\mathsf{Judge}(\mathsf{gpk}, \mathsf{i}, \mathsf{info}_\tau, \pi_{\text{Trace}}, \mathbf{upk}[\mathsf{i}], m, \Sigma)} \to 1/0$
- Return $\mathsf{ARS}_{Judge}(\mathsf{tpk}, m||\tau, R_\tau, \Sigma, \mathbf{upk}[\mathsf{i}], \pi_{\text{Trace}})$.

$\underline{\mathsf{IsActive}(\mathsf{info}_\tau, \mathsf{reg}, \mathsf{i})}$
- Check info_τ to see if user i is in R_τ, and returns $0/1$ accordingly.

Figure 7.7: Construction of a fully dynamic group signature from an accountable ring signature

bound on the number of UToM queries $\mathcal{A}$ can make in her game. Adversary $\mathcal{B}$ randomly chooses an index $i \leftarrow \{1,\ldots,q\}$ and sets the challenge key pair (vk, sk) of his game as the keys of the i-th user i^* queried to the UToM oracle: $(\mathbf{usk}[\mathsf{i}^*], \mathbf{upk}[\mathsf{i}^*]) := (sk, vk)$. By construction, user keys do not depend on i, so the simulation is perfect. Whenever $\mathcal{A}$ succeeds in breaking correctness by returning (i, m, τ), $\mathcal{B}$ succeeds when $\mathsf{i} = \mathsf{i}^*$ by returning $(\mathsf{tpk}, m||\tau, R_\tau)$. Note that i is independent of the success probability of $\mathcal{A}$. Hence, the success probability of $\mathcal{B}$ is that of $\mathcal{A}$ divided by a factor of q, and thus correctness of the fully dynamic group signature scheme follows.

For anonymity, suppose that $\mathcal{A}$ is an adversary against the anonymity of the group signature scheme. We construct an adversary $\mathcal{B}$ against the anonymity of the accountable ring signature scheme. Adversary $\mathcal{B}$ gets $\mathsf{crs} := pp$ and pk from his anonymity game. He sets $\mathsf{tpk} := pk$ and interacts with $\mathcal{A}$ on behalf of the tracing manager of the fully dynamic group signature scheme to get $(\mathsf{msk}, \mathsf{mpk}, \mathsf{info})$. Note that $\mathcal{B}$ does not know dk corresponding to pk and hence does not know the secret tracing key tsk of the tracing manager. $\mathcal{B}$ now sets $\mathsf{gpk} := (\mathsf{crs}, \mathsf{mpk}, \mathsf{tpk})$ and initialises reg. For all oracles except Trace and Chal_b, $\mathcal{B}$ is able to simulate all queries of $\mathcal{A}$ by himself and return the result, since $\mathcal{B}$ knows all necessary keys. If $\mathcal{A}$ queries $(m, \Sigma, \mathsf{info}_\tau)$ to her Trace oracle, $\mathcal{B}$ can simulate the oracle by querying $(m||\tau, R_\tau, \Sigma)$ to his Open oracle and forwards the answer to $\mathcal{A}$. If $\mathcal{A}$ queries $(\mathsf{info}_\tau, \mathsf{i}_0, \mathsf{i}_1, m)$ to her Chal_b oracle, $\mathcal{B}$ simulates the oracle by querying $(m||\tau, R_\tau, \mathbf{gsk}[\mathsf{i}_0], \mathbf{gsk}[\mathsf{i}_1])$ to his Chal_b oracle and returns the answer to $\mathcal{A}$. Eventually, when $\mathcal{A}$ outputs her guess b^*, $\mathcal{B}$ returns b^* as his answer in his game. It is clear that both adversaries have the same advantage. Therefore, by anonymity of the accountable ring signature scheme, the fully dynamic group signature scheme satisfies our full anonymity definition.

Next, we prove non-frameability. Let $\mathcal{A}$ be an adversary against the non-frameability of the fully dynamic group signature scheme. We construct an adversary $\mathcal{B}$ attacking the full-unforgeability of the accountable ring signature scheme. Given pp, $\mathcal{B}$ starts $\mathcal{A}$ on $\mathsf{crs} := pp$ and receives $(\mathsf{info}, \mathsf{msk}, \mathsf{mpk}, \mathsf{tsk}, \mathsf{tpk})$ using which $\mathcal{B}$ can compute gpk which he forwards to $\mathcal{A}$. Now, for all oracles except SndToU, Reveal and Sign, $\mathcal{B}$ is able to simulate all queries of $\mathcal{A}$ by himself and return the result. For the other three oracles, $\mathcal{B}$ can simulate those successfully by consulting his own oracles $\mathrm{UKGen}, \mathrm{Sign}$ and RevealU if necessary. Eventually, when $\mathcal{A}$ outputs $(m, \Sigma, \mathsf{i}, \pi_{\mathrm{Trace}}, \mathsf{info}_\tau)$, $\mathcal{B}$ returns $(\mathsf{tpk}, \mathbf{upk}[\mathsf{i}], m||\tau, R_\tau, \Sigma, \pi_{\mathrm{Trace}})$ in his game. Clearly, if $\mathcal{A}$ wins her game, $\mathcal{B}$ succeeds in breaking the full unforgeability of the accountable ring signature since his output satisfies the first clause of the winning condition.

Next, we prove traceability. Let $\mathcal{A}$ be an adversary against the traceability of the fully dynamic group signature scheme. We construct an adversary $\mathcal{B}$ against the traceability of the accountable ring signature scheme. Given pp, $\mathcal{B}$ sets $\mathsf{crs} := pp$, generates $(\mathsf{msk}, \mathsf{mpk})$ and initialises reg and info. He interacts with $\mathcal{A}$ on behalf the group manager in order to obtain $(\mathsf{tsk}, \mathsf{tpk})$. He sets $\mathsf{gpk} := (\mathsf{crs}, \mathsf{mpk}, \mathsf{tpk})$ and forwards $(\mathsf{gpk}, \mathsf{info})$ to $\mathcal{A}$. Now, $\mathcal{B}$ is able to simulate all of the oracle queries of $\mathcal{A}$. Eventually, when $\mathcal{A}$ outputs (m, Σ, τ) where Σ is a valid signature, $\mathcal{B}$ returns $(\mathsf{tsk}, m||\tau, R_\tau, \Sigma)$ in his game. We have three cases to consider depending on how $\mathcal{A}$ wins her game. If i is inactive group member at epoch τ, then $\mathsf{i} \notin R_\tau$. If $\mathsf{i} = 0$ then again $\mathsf{i} \notin R_\tau$. The final case is that the group signature judge algorithm returns 0. The probability of success of $\mathcal{B}$ is the same as that of $\mathcal{A}$. Thus, by the traceability of the accountable ring signature scheme, the fully dynamic group signature scheme satisfies traceability.

Finally, we prove tracing-soundness. Let $\mathcal{A}$ be an adversary against the tracing-soundness of the fully dynamic group signature scheme. We construct an adversary $\mathcal{B}$ against the tracing-soundness of the accountable ring signature scheme. Given pp, $\mathcal{B}$ initialises reg and sets $\mathsf{crs} := pp$. He starts $\mathcal{A}$ on crs to get $(\mathsf{info}, \mathsf{msk}, \mathsf{mpk}, \mathsf{tsk}, \mathsf{tpk})$. He sets $\mathsf{gpk} := (\mathsf{crs}, \mathsf{mpk}, \mathsf{tpk})$ which he then forwards to $\mathcal{A}$. Note that all $\mathcal{A}$'s oracle calls can be simulated by $\mathcal{B}$ since he has the required keys. Eventually, $\mathcal{A}$ halts by responding with a tuple $\left(m, \Sigma, \mathsf{i}_1, \pi_{\text{Trace}_1}, \mathsf{i}_2, \pi_{\text{Trace}_2}, \mathsf{info}_\tau\right)$. By construction, Σ returned by $\mathcal{A}$ is a valid accountable ring signature on m that traces to two different users. Adversary $\mathcal{B}$ returns $\left(m||\tau, \Sigma, \mathsf{tpk}, \mathbf{upk}[\mathsf{i}_1], \mathbf{upk}[\mathsf{i}_2], \pi_{\text{Trace}_1}, \pi_{\text{Trace}_2}\right)$ as his output in his game. Clearly, if $\mathcal{A}$ wins her game, $\mathcal{B}$ wins his game with the same advantage. Thus, if the accountable ring signature scheme satisfies tracing soundness so does the fully dynamic group signature scheme. $\square$

Chapter 8

Helios Ballot Copying Revisited

In this chapter and the next we will develop two different answers to the problem of voting receipts. Intuitively, a receipt is information that enables a voter to fully reproduce her ballot as posted in an election's bulletin board. This can lead to issues with coercion, as a voter who can take possession of a receipt can be bribed in a verifiable way.

In the case of Helios [Adi08], the protections against receipts (and coercive attacks in general) are minimal. The code running the voting booth aims to prevent receipts by allowing voters to either audit or cast a ballot. If a ballot is audited, the vote and randomness used is revealed (so that it can be checked later), but the voter cannot cast it. If it is cast, the the randomness is not displayed to the voter. However, in the usual case, Helios would be deployed as a web application which users would use by visiting the election web page using their personal computers. Thus, a user who is technically proficient, (or who is given ready-built tools) is able to circumvent the "cast or audit" restriction. In fact, it is possible to both extract and inject ballot parameters from and into the voting booth application.

As such, the Helios voting system suggests that it should not be used for elections where the risk of coercion is high. Thus, the simple answer to voting receipts is to try and combat them. In this chapter however, we develop a different possibility: we use the divertibility of the zero knowledge protocols used in Helios to build a secure ballot copying protocol.

Cortier & Smyth [CS11] explored ballot copying in the Helios e-voting platform as an attack against privacy. They also pointed out that their approach to ballot copying could be detected by a modified Helios. In an earlier version of this work, [DC12] revisited ballot copying from a different viewpoint: as a tool to prevent vote diffusion (the division of votes among multiple weak candidates) and to lessen the effect of established voting blocs. This approach is based on *blinding* the ballot casting protocol to create an undetectable copy. A willing voter can cooperate with a prospective copier, helping the copier produce a blinded copy of his ballot without revealing his vote. We prove that Helios is unable to detect the copying. The possibility of such cooperation between voters is manifested only in internet voting and as such is a fundamental difference between internet and booth voting.

While the techniques of [DC12] are sound, there are two outstanding issues. First, the copying protocol gives zero knowledge only against honest copiers unless we opt to break compatibility with Helios. Second, as Smyth and Bernhard [SB13] point out, the (informal) security guarantees can be misleading. In particular, the act of producing a copy may reveal information about one's vote via the election's result even if the copying protocol itself is zero knowledge. We correct both issues, producing a fully zero knowledge protocol for ballot copying without breaking compatibility with Helios.

The research presented in this chapter is joint work with Yvo Desmedt, and significantly updates work published in [DC12]. The author contributed to the investigation by designing the updated, full zero knowledge, protocol and providing the appropriate proofs as well as updating the discussion on privacy.

8.1 Introduction

Helios [Adi08] is a web-based, universally verifiable Internet voting system. To facilitate universal verifiability, Helios ballots are encrypted and public. Ballots are cast over the Internet via a web browser. By way of controlling their web browser, users have full control over the ballots they submit (see Sect. 8.2.1), making them susceptible to coercion. If a voter is bribed or threatened, they can be instructed to vote in a particular way and keep the randomness used as a receipt. This way, a potential coercer can check if they complied or not. Helios is thus best suited for use in low-coercion environments. This might appear to limit its use to "low-stakes" elections but may not necessarily be the case. In Estonian parliamentary elections [Est10] for example, the ability of overwriting a vote with a later one is used in place of coercion resistance. As such, it is conceivable that a system like Helios might be used in a high profile election.

Even though Helios has been based on 30 years of sound cryptographic primitives, previous works have described attacks against Helios compromising both secrecy [CS11] and correctness [ED10]. Furthermore, the exact security achieved by the Helios construction was the subject of active research [BPW12, SB13]. Other systems [RRI15, GCG15] build upon Helios with the intention of improving its security or robustness.

Our work here has a different goal though: blinded ballot copying. A *blinded* copy of a ballot is a copy that cannot be detected as such. Instead of a forced relationship between coerced and coercer, this form of ballot copying relies on the cooperation of both parties, and is based on trust rather than threats or bribes. This demonstrates how the lack of receipt freeness hides an unspecified property of Helios: the ability to create blinded copies of votes. In fact, it can be expanded upon to build a secondary system on top of it. The potential for this was also mentioned in [BGP11], independently of [DC11].

Assume that Alice and Bob are coworkers. Clint and Donnie are candidates for the "employee of the year award". Bob has recently returned from a project abroad and is unsure about the two candidates. He would like to ask Alice whom he trusts. Alice does not want to reveal her choice so as not to upset the other candidate. Our goal is to provide a system where Alice (the voter) can assist Bob (the copier) in producing a copy of her ballot whilst ensuring that:

- Bob will not learn anything about Alice's vote that is not also revealed by the election results, and his possessing of a copy of her vote.

- Alice cannot distinguish the ballot that is produced by the system from a random valid ballot. Therefore, the copier is explicitly given the option of backing out (by using his own choice instead of the copied one) undetected.

- Helios (or any observer) cannot recognise the ballot produced by the system as a copy.

Such a system would allow groups of voters to organise around a trusted figure, partly avoiding the spoiler effect [Arr63] prevalent in plurality elections, thus increasing the weight of their vote and the possibility of obtaining a desired result (to the degree where trust in the original voter is well-deserved).

However, increasing the weight of one's vote carries privacy implications[1]. If Bob copies Alice's vote and Clint only gets a single vote in the tally, then Alice's vote is revealed. This parallels the discussion of privacy in the event of a unanimous election. The privacy definitions used in the area (e.g. Def.21) overcome this problem by running two bulletin boards: a real one (which is properly tallied) and a fake one (which is not, and only contains null ballots). The adversary interacts with the real board, submitting the vote preferences of honest users or potentially maliciously crafted ballots for corrupt ones. The interactions are mirrored in the fake board, but honest votes are replaced with zeros. The challenge for the adversary is to correctly guess which board is visible to him during the experiment. Thus, to maintain privacy, ballot copying needs to be able to copy the ballots that exist in the real board irrespective of which board is visible.

8.2 Background

The work presented in this chapter is based on modifying the protocols used by Helios so that they involve three parties (voter, copier, Helios) instead of just two. As such, we need to explain the design and operation of Helios before describing the modifications.

An important feature implemented by Helios is universal verifiability [SK95,CGS97b]: any party, even one uninvolved in the election can opt to verify the integrity of an election that uses Helios. This is achieved by making the ballots cast by each voter publicly accessible on a bulletin board, albeit encrypted with the ElGamal [Elg85] cryptosystem.

Each ballot also contains a proof of its validity which can be verified without requiring specific knowledge or access and without revealing the contents of the ballot. The proofs of

[1]We are thankful to Ben Smyth and David Bernhard for pointing out [SB13] that the original wording about Alice's privacy was too strong.

validity are based on a disjunctive version of the Chaum-Pedersen protocol [CDS94, CP93] previously used in [CGS97b]. This ensures that no invalid ballots have been accepted and that no ballot tampering has taken place. The public list of ballots also guards against the election officials injecting votes from unregistered voters if the registration list is public. As the encryption scheme used is additively homomorphic, the product of all encrypted votes is an encryption of the sum of all votes. Since the encrypted ballots are all public, there is no way for a corrupt server to tamper with the product in an undetected way. The vote sum is obtained by the election trustees using threshold decryption. Each trustee is able to provide a partial decryption factor along with a proof of correctness for his individual calculations. The partial decryption factors are then combined to arrive at the decrypted result. Again, once the partial decryption factors have been made public there is no opportunity for foul play.

In this section we will analyze the parts of Helios that are relevant to this work. We will start by briefly mentioning the relevant parts of the Helios Implementation before moving to the cryptographic design. The design of Helios 3.5 is based on the ElGamal cryptosystem [Elg85], used for encrypting votes and homomorphic tallying. It also uses disjunctive zero-knowledge proofs of equality to ensure ballot validity.

8.2.1 Helios Implementation

As mentioned in [Hel11] Helios has 4 main components: an election builder, a voting booth, a ballot casting server and an audit server. From the perspective of ballot copying, we are mostly concerned with the inner workings of the voting booth since we need to be able to extract data in order to capture the encryption randomness and also inject it to allow the copied and blinded ballot to be actually submitted. The ballot casting server concerns us only with respect to the tests performed against incoming ballots, according to the Helios specifications [Hel11]. The workings of the other two components are not relevant.

VOTING BOOTH. The Helios voting booth is a fairly complex web application, that reads the parameters of an election, presents the user with the questions he can vote on, encrypts his choices and calculates the appropriate proofs to construct a valid ballot and then allows the user to either audit *or* submit it.

The voting booth in Helios 3 is implemented in HTML and JavaScript. For performance reasons, older versions of the voting booth required Java to perform the modular arithmetic operations required in the encryption and proof construction, while keeping the encryption logic in JavaScript. Another option added later was for the ballot construction to be performed on the server and the results forwarded to the browser. However, more recent versions of Helios include a high performance JavaScript library for handling large integers, removing the need for external plugins, or server-side calculations.

The randomness used in the encryption is only revealed if the user chooses to audit his ballot in which case he will need to create a new one before voting. This is implemented as a weak form of coercion resistance, (namely, receipt freeness) but is easily bypassed[2] if the voter executes a JavaScript command during the preparation of the ballot.

VOTING BOOTH TAMPERING. Web browsers allow users to run their own javascript commands, to be executed in the context of the currect web-page. Thus, a user can simply cause the browser to output the randomness as well as the vote contained in a ballot, just before it is cast. Since the commands are executed in the full context of the web-page one can take advantage of the functions provided by the voting booth, e.g. calling `BOOTH.encrypted_ballot.get_audit_trail();` instead of manually reading the ballot parameters. Similarly, if a voter is given a pre-made ballot by a coercer, they only need to overwrite the existing data structure representing their ballot with the pre-made one.

Executing JavaScript commands might be difficult for the average user, but in this instance, the commands in question may be stored as "bookmarklets". A bookmarklet is a bookmark that contains a javascript code instead of a location. When it is clicked, the code is run in the context of the page currently displayed. For Helios, it is simple to construct a bookmarklet that extracts the vote and randomness used to create a ballot (i.e a receipt) as well as one that injects a specific ballot to the voting booth page. Thus a user only needs the technical expertise to install a bookmarklet (i.e drag & drop it to their bookmark bar) and click it at the appropriate time in order to tamper with the voting booth.

We note that the techniques described in the rest of this Chapter do not represent an attack on the helios Voting Booth, or new vulnerabilities to be abused: the potential for receipt extraction and ballot injection is inherent in the current implementation of Helios and browser environment.

BALLOT CASTING SERVER. After the ballot is constructed and saved as a JSON (JavaScript Object Notation) [Cro06] object, the voting booth submits it to the ballot cast-

[2]In fact, early versions of Helios included a "Coerce Me!" button [Adi08] which revealed the encryption randomness without invalidating the ballot.

ing server using the HTTP POST method. The ballot casting server then checks the ballot for validity and compares it against already cast ballots and rejects it, if it is identical to a previous one.

8.2.2 Vote Representation

In Helios, the homomorphic property of ElGamal (Sect.3.3.5) is used in order to calculate an encrypted *sum* of votes from individual encrypted ballots without the need to decrypt them individualy. However, ElGamal is multiplicatively homomorphic, whereas vote tallies are sums. In order to bridge this gap, Helios uses a variant of the encoding used in [CGS97b]. where votes of "no" are represented as 1 and votes of "yes" as g. In this way, the product of n votes v_i of which m are "yes" will be $\prod_{i=1}^{n} v_i = g^m$ i.e. the log of the product will be the sum of the votes i.e. the scheme is additively homomorphic. Most elections have more options than "yes" and "no", so Helios models them as a series of "yes-no" questions about each option, with a limit on the number of "yes" answers equal to the number of selections allowed in the original question. For example, given a question with 3 choices, from which exactly one may be selected, a vote would be of the form:

$$\begin{aligned} V &= (\alpha_0, \beta_0), (\alpha_1, \beta_1), (\alpha_2, \beta_2) \\ &= (g^{r_0}, g^{m_0} \cdot h^{r_0}), (g^{r_1}, g^{m_1} \cdot h^{r_1}), (g^{r_2}, g^{m_2} \cdot h^{r_2}) \end{aligned}$$

In the above vote, r_i represents the randomness used in the encryption and m_i the answers of the voter to each of the 3 options. We note that a vote of the above form might be *invalid*, for example if $m_i > 1$ for some i, or if every m_i is 1, even though the election parameters only allow the voter to choose one option. A particularly insidious voter might even have $m_0 = -100$, making his vote cancel out 100 honest votes for the first option. Helios guards against this by requiring the voter to provide a *zero-knowledge* proof of his ballot's validity. Helios supports threshold ElGamal, which involves multiple trustees in order to ensure that there is no single point of failure with regard to voter privacy, but this is not relevant to this work.

8.2.3 Validity Checks

As seen in the above example, a voter must provide a proof that the value of his vote falls into the range permitted by the election parameters. As such, she must prove that the individual vote for each option is either a "yes" or a "no" and furthermore that the total number of "yes" votes is within the range of the allowed number of selections. In more concrete terms, the voter is asked to prove that each m_i is either 0 or 1 (an *individual proof* in Helios terminology), and that the sum $\sum_{i=0}^{n-1} m_i$ is inside the range of allowed selections as specified in the election's definition (a *total proof*).

The checks of validity used by Helios are non-interactive disjunctive zero knowledge proofs of equality between discrete logs. In the rest of this section, we will offer a brief overview of the underlying concepts as well as their use in Helios.

8.2.4 Disjunctive Proofs of Equality between Discrete Logarithms.

To prove that an encrypted individual vote (α, β) is valid one must prove that the corresponding plaintext is either $g^0 = 1$, in which case $\log_g \alpha = \log_h \beta$, or g^1 in which case $\log_g \alpha = \log_h \beta / g$. As the prover needs to prove the *disjunction* of the two statements we have a *disjunctive proof.*

Total proofs can be carried out in the same way, the difference being that for individual proofs the range of exponents is always $[0,1]$ whereas for total votes it ranges from the minimum number of selections to the maximum. For total proofs the ciphertext used is the homomorphic product of the individual ciphertexts.

The Chaum-Pedersen protocol [CP93] for discrete log equality is essentially a parallel version of the Schnorr protocol [Sch91]. We note that the Chaum-Pedersen protocol as well as the underlying Schnorr protocol are Σ-protocols and this only achieve honest verifier zero knowledge. There is no known simulator for dishonest adversaries [CP93]. In regular usage this is sidestepped using the Fiat-Shamir heuristic.

Let g, h be elements of $\mathcal{G}_{group}$ of order q. Suppose $\alpha = g^w, \beta = h^w$. A prover with private input w is trying to prove that $\log_g \alpha = \log_h \beta$. The protocol is as follows:

Protocol 1. *Equality of Discrete Logs [CP93]*

Prover: *Let $a := g^r$ and $b := h^r$ for $r \leftarrow \mathbb{Z}_p$. Send a, b to the Verifier.*

Verifier: *Choose $c \leftarrow \mathbb{Z}_p$. Send c to the Prover.*

Prover: *Let $s := cw + r$. Send s to the Verifier.*

Verifier: *Accept if $g^s = a\alpha^c$ and $h^s = b\beta^c$.*

Cramer et al. provide a construction for disjunctive proofs [CDS94, FS90] where the prover can prove one statement from a set and simulate proofs for the other ones, without the verifier knowing which of the subproofs are simulated. In the context of Helios, this allows the voter to indicate that the plaintext of his ballot is one out of a number of allowed values without revealing which one.

In order to apply the construction of Cramer et al. [CDS94] to the Chaum-Pedersen protocol [CP93], we use the simulator to produce the proofs corresponding to the false statements:

Protocol 2. *Chaum-Pedersen Protocol Simulation*

Simulated Proof*: Choose random challenge c and response s. Let the commitments be $a = g^s/\alpha^c$ and $b = h^s/(\beta/g^v)^c$.*

Verification*: Check that $g^s = a \cdot \alpha^c$ and that $h^s = b \cdot (\beta/g^v)^c$.*

In order to force the voter to provide at least one honest proof, he is not given complete choice of the challenges. The Verifier is allowed to specify the sum of the challenges used in the subproofs. This allows the voter to simulate all but one of the Chaum-Pedersen proofs and let the challenge of the real subproof be as a balancing factor in the sum. Suppose the voter needs to prove that the value v encoded by $(\alpha, \beta) = (g^r, h^r g^v)$ is in $[\min, \max]$. He will simulate the proofs for $i \in [\min, \max] \setminus \{v\}$ and produce a real proof for $i = v$.

Protocol 3. *Disjunctive Chaum-Pedersen Protocol.*

Prover*: For $i \in [\min, \max] \setminus \{v\}$: Choose challenge $c_i \leftarrow \mathbb{Z}_p$ and response $s_i \leftarrow \mathbb{Z}_p$. Let the commitments be $a_i := g^{s_i}/\alpha^{c_i}$ and $b_i := h^{s_i}/(\beta/g^i)^{c_i}$. Let $(a_v, b_v) = (g^w, h^w)$ for $w \leftarrow \mathbb{Z}_p$. Send (a_i, b_i), for $i \in [\min, \max]$.*

Verifier*: Send $x \leftarrow \mathbb{Z}_p$.*

Prover*: Let $c_v := x - \sum_{i \neq v} c_i$ and $s_v := rc_v + w$. Send (c_i, s_i), for $i \in [\min, \max]$.*

Verifier*: Check if $g^{s_i} = a \cdot \alpha^{c_i}$ and that $h^{s_i} = b \cdot (\beta/g^i)^{c_i}$ for $i \in [\min, \max]$. Check if $x = \sum_{i=min}^{max} c_i$. Accept only if all checks passed.*

8.2.5 Non-Interactive Proofs.

For practical reasons, Helios implements the above protocol offline rather than online. This requires less communication with the Helios server and does not require the Helios server to hold the state of proof protocols in progress. This is done by way of the Fiat-Shamir heuristic (Sect. 3.4.6) which replaces the random challenge issued by the verifier with a hash of the commitments.

This facilitates universal verifiability since the generation of the challenge is beyond the control of the (potentially dishonest) Helios server. Additionally, the entirety of the ballot generation can be performed by the voter with the result being submitted to Helios for verification. This eliminates the need for the server to hold state between multiple sessions, which can have performance as well as security implications.

For example, suppose we have an election with 3 options of which exactly one may be selected (as in the vote example). We follow the notation of protocol 3 in that a_i, b_i represent the commitments of a proof, s_i the solution and c_i the challenge. The individual proofs would then be of the form:

$$\pi_i = ((a_{i,0}, b_{i,0}, a_{i,1}, b_{i,1}), (c_{i,0}, s_{i,0}, , c_{i,1}, s_{i,1})), \text{ for } i \in \{0,1,2\} \tag{8.1}$$

And the total proof would be of the form:

$$\pi_\Sigma = ((a_{\Sigma,1}, b_{\Sigma,1}), (c_{\Sigma,1}, s_{\Sigma,1}))$$

To check if one of the above proofs is valid, Helios would set $T := H(a_{\min}, b_{\min}, \cdots, a_{\max}, b_{\max})$ and run the last step of protocol 3.

In the interest of readability, for the rest of this text we will limit ourselves to ballots consisting of a single encrypted “yes”-“no” vote and it’s corresponding proof of validity.

Blind signatures [Cha83, CP93], involve signing a document through an intermediary (in our case, the copier) without the original signer (the voter) being able to trace the end product. Blind signatures have been suggested by Chaum [Cha83] for use with anonymous electronic cash, where banks sign “coins” proving their authenticity but are unable to trace their use, and voting where authorities can supply signed blank ballots to authenticated voters but are then unable to track them once filled.

Divertible proofs [OO90, DGB88] are a similar notion to blind signatures, but in an online setting. An intermediate party is introduced between the prover and verifier, playing the role of the verifier against the prover and that of the prover against the verifier. The intermediate is called a *warden* in some cases (for example, if he is introduced to enforce to ensure honest behaviour) or a man in the middle in others. In this chapter we will refer to the warden as the copier.

8.3 A Ballot-Blinding Protocol

We will describe ballot-blinding in two parts. We will first describe *vote-blinding*, i.e. how a copier can unilaterally rerandomise the vote contained in a ballot (making it indistinguishable to a random one assuming the DDH problem is hard) while performing appropriate modifications to keep the attached proofs valid. We note that this is contingent on Helios using the weak [BPW12] version of the Fiat-Shamir transformation (i.e. the challenge is produced as a hash of the prover's first message and does not depend on the statement). Since this rerandomization does not change the prover's first message or the challenges used inside the proofs, it is trivial to detect.

On the other hand, the non-malleability [FKMV12] of the Fiat-Shamir transformation does not give us much hope of unilaterally randomizing the proofs. A trivial solution would be for the voter to publish the randomness used in her ballot. This would enable blinded copying but would would also completely sacrifice her privacy! For that reason, we need an online *proof-blinding* protocol between a willing voter (who has already cast a ballot) and a copier. The protocol allows the copier to produce a "new" proof of knowledge for the encrypted vote.

The copier can combine the two parts: first he obtains a new (indistinguishable) proof of validity of the voter's encrypted vote and then he re-encrypts the encrypted vote making it indistinguishable as well. The result is a ballot that is equivalent to the original in that it contains the same vote but indistinguishable from it. Moreover, it does not leak the original vote. In the event that Helios moves to use the strong Fiat-Shamir transformation, the copier would need to perform both blinding operations concurrently. As this is not currently the case, we opt to present the two operations separately for simplicity.

8.4 Vote Blinding

We describe a transformation that a copier can perform to an already cast ballot that is based on re-encrypting the vote contained in the ballot. Because of the re-encryption, the proof contained in the ballot must also be modified to stay valid.

Given a vote $(\alpha, \beta) = (g^r, h^r g^v), v \in 0, 1$, a copier is able to re-encrypt it as $(\alpha', \beta') = (g^{r+z}, h^{r+z} g^v), v \in 0, 1$. To do that, he does not need knowledge of r as he can simply calculate $(\alpha', \beta') = (g^z\alpha, h^z\beta)$.

Lemma 11. *If z is chosen to be uniformly random in $\mathbb{Z}_p$ then (α', β') is indistinguishable from a random vote by adversaries who cannot solve the DDH problem, regardless of them knowing r or v.*

Proof. Suppose we have an adversary $\mathcal{A}$ who can, given (g,h) and $(\alpha,\beta),(\alpha',\beta')$ determine if (α',β') is a re-encryption of (α,β) or not. Equivalently, $\mathcal{A}$ can determine if $(\alpha'/\alpha, \beta'/\beta)$ is an encryption of 0, i.e of the form g^σ, h^σ. As $h = g^x$, $(h, \alpha'/\alpha, \beta'/\beta)$ is a DH tuple. This enables us to build a DDH solver as follows:

Given a DDH instance (gk, h, u, v) where $gk = (\mathcal{G}, q, g)$, pick random $\rho, \psi \leftarrow \mathbb{Z}_q$, set $(\alpha,\beta) := (g^\rho, h^\rho \cdot g^\psi)$, and let $(\alpha', \beta') := (u \cdot \alpha, v \cdot \beta)$. Give g, h and $(\alpha,\beta),(\alpha',\beta')$ to the vote distinguisher $\mathcal{A}$, accept the tuple if $\mathcal{A}$ accepts and reject it if it rejects.

□

Furthermore, if the copier has access to a valid proof for (α,β) he can transform it to a valid proof for (α',β').

Lemma 12. *If (V,P) is a valid ballot, with $V = (\alpha,\beta)$ as in (8.1) and $P = ((a_0,b_0,a_1,b_1),(c_0,s_0,c_1,s_1))$ as in Protocol 3, then $((g^z\alpha, h^z\beta),(a_0,b_0,a_1,b_1),(c_0,s_0 + c_0z, c_1, s_1 + c_1z))$ is also a valid ballot and vice versa.*

Proof. If $a_i = g^{s_i}/\alpha^{c_i}$ holds then $a_i = g^{s_i+c_iz}/(g^z\alpha)^{c_i}$ also holds. Similarly, if $b_i = h^{s_i}/(\beta/g^i)^{c_i}$ holds then $b_i = h^{s_i+c_iz}/(h^z\beta/g^i)^{c_i}$ also holds. For the opposite direction we note that re-applying the transformation for $-z$ produces the original ballot. □

The above transformation can be used as a variant of the attack described in [CS11] since it provides another way of replaying ballots without copying them verbatim. Nonetheless, the attack variant can be stopped in a similar way to the one suggested by Cortier and Smyth, i.e by checking the commitments a_i, b_i and challenges c_i of the proof, which remain unchanged. Thus, a future version of Helios could defend against the attack by modifying the ballot casting server to reject votes which reuse past commitment values.

8.5 Proof Blinding

It is clear from the above discussion that blinding the entire ballot is necessary. However, proof blinding protocol requires two assumptions: First, that original voter cooperates with the the copier and second, that the voter has access to the randomness used in encrypting his ballot (i.e a receipt). Fortunately, the second assumption can be fulfilled in the current Helios implementation. At the same time, since the active consent of the original voter is required, our work is of no use to an attacker. If the attacker is able to force the original voter to reveal their randomness, he is directly violating their privacy. Similarly, if an attacker is able to force somebody to submit a "sealed" ballot, sight-unseen, there is not need for a copying protocol to effectively steal their vote.

We start by presenting the protocol of [DC12] for reference before moving on to our version.

8.5.1 The proof blinding protocol of [DC12]

Suppose the voter has cast an encrypted vote $(\alpha,\beta)=(g^r,h^rg^v), v\in 0,1$ with an appropriate proof, and the copier is requesting a different proof in order to copy it. Note that (α,β) is public but (r,v) is private to the voter. Since the hash function H() is public, Helios does not take part in the protocol. The notation used for the commitments is the same as in Protocol 3, but the roles of the parties are different. The voter still takes the role of the prover, but the copier takes the role of an intermediate verifier who ultimately submits the resulting ballot to Helios.

Protocol 4. *Proof Blinding Protocol*

Voter *Choose $w\leftarrow\mathbb{Z}_p$ and let $a_v := g^w, b_v := h^w$. Let $\phi := 1-v$ and choose $c_\phi, s_\phi \leftarrow \mathbb{Z}_p$ and let $a_\phi := g^{s_\phi}/\alpha^{c_\phi}$ and $b_\phi := h^{s_\phi}/(\beta/g^\phi)^{c_\phi}$.*

Send (a_0,b_0,a_1,b_1) to the copier.

Copier *Choose $\Delta_0,\Delta_1,k_0,k_1 \leftarrow \mathbb{Z}_p$. Let $A_i := a_ig^{k_i}/\alpha^{\Delta_i}$, $B_i := b_ih^{k_i}/(\beta/g^i)^{\Delta_i}$ for $i=0,1$. Let $c := H(A_0,B_0,A_1,B_1)$ be the challenge that Helios would issue. Let $C := c-\Delta_0-\Delta_1$.*

Send C to the voter.

Voter *Let $c_v := C-c_\phi$ and let $s_v := w+rc_v$.*

Send (c_0,s_0,c_1,s_1) to copier as a reply.

Copier *Check that $C=c_0+c_1$, $a_i=g^{s_i}/\alpha^{c_i}$ and $b_i=h^{s_i}/(\beta/g^i)^{c_i}$ for $i=0,1$. If yes, accept and let $C_i := c_i+\Delta_i$ and $S_i := s_i+k_i$. Let $V := ((\alpha,\beta),(A_0,B_0,A_1,B_1),(C_0,S_0,C_1,S_1))$ and submit V to Helios. Otherwise, reject.*

While the protocol is complete and sound, it only achieves honest verifier zero knowledge; [DC12] also features a full zero knowledge alternative, but at the cost of breaking compatibility with Helios. This is problematic in itself, but also does little to help us with constructing a security reduction that enables us to simulate copying one ballot while actually copying another.

8.5.2 New Proof Blinding Protocol

We suggest a simple addition to protocol 4 that not only gives full (computational) zero knowledge, but also requires no additional rounds of interaction or changes to Helios. Furthermore, our fix is fairly general and could be used in other applications where a HVZK proof is to be diverted. The main idea is to allow the prover to change the statement but only after proving it is equivalent to the original. In normal use, this is inconsequential, and the additional proof means that the change does not impact soundness. At the same time, if the simulator has a trapdoor for the equivalence proof (in our case this means programming the random oracle) it is simple to obtain zero knowledge by making the new statement independent of the original.

This not only enables us to prove full zero knowledge but also gives us an important tool for ballot privacy: the adversary can ask to copy ballot $\mathcal{B}$ and the reduction can use statement switching to provide him with $\mathcal{B}'$.

Suppose the voter has cast an encrypted vote $(\alpha, \beta) = (g^r, h^r g^v), v \in 0,1$ with an appropriate proof, and the copier is requesting a different proof in order to copy it. Note that (α, β) is public but (r, v) is private to the voter.

Protocol 5. *Proof Blinding Protocol*

Voter *Choose $\tilde{r} \leftarrow \mathbb{Z}_p$, let π be a non-interactive Chaum-Pedersen proof that $(\tilde{\alpha}, \tilde{\beta}) = (g^{\tilde{r}}, h^{\tilde{r}})$ have the same discrete log w.r.t (g,h). Let $\alpha' = \alpha\tilde{\alpha}$, $\beta' = \beta\tilde{\beta}$ and $r' = r + \tilde{r}$.*

Choose $w \leftarrow \mathbb{Z}_p$ and let $a_v := g^w, b_v := h^w$. Let $\phi := 1 - v$ and $c_\phi, s_\phi \leftarrow \mathbb{Z}_p$ and let $a_\phi := g^{s_\phi}/\alpha'^{c_\phi}$ and $b_\phi := h^{s_\phi}/(\beta'/g^\phi)^{c_\phi}$. Send $(\tilde{\alpha}, \tilde{\beta}, \pi), (a_0, b_0, a_1, b_1)$ to the copier.

Copier *Reject if π does not verify. Let $\alpha' = \alpha\tilde{\alpha}$, $beta' = \beta\tilde{\beta}$. Choose $\Delta_0, \Delta_1, k_0, k_1 \leftarrow \mathbb{Z}_p$. Let $A_i := a_i g^{k_i}/\alpha'^{\Delta_i}$, $B_i := b_i h^{k_i}/(\beta'/g^i)^{\Delta_i}$ for $i = 0,1$. Let $c := H(A_0, B_0, A_1, B_1)$ be the challenge that Helios would issue. Let $C := c - \Delta_0 - \Delta_1$, send C to the voter.*

Voter *Let $c_v := C - c_\phi$ and let $s_v := w + r'c_v$, send (c_0, s_0, c_1, s_1) to the copier.*

Copier *Check that $C = c_0 + c_1$, $a_i = g^{s_i}/\alpha'^{c_i}$ and $b_i = h^{s_i}/(\beta'/g^i)^{c_i}$ for $i = 0,1$. If yes, accept and let $C_i := c_i + \Delta_i$ and $S_i := s_i + k_i$. Let $V := ((\alpha', \beta'), (A_0, B_0, A_1, B_1), (C_0, S_0, C_1, S_1))$*

and send V to Helios. Otherwise, reject.

We will now examine Protocol 5 with regard to correctness, soundness, indistinguishability and zero knowledge.

Lemma 13. *The new proof blinding protocol is complete and furthermore if an honest copier accepts then the resulting ballot V will be accepted by Helios.*

Proof. Completeness holds trivially. Indeed, we have:

- $C = c_0 + c_1$ since the voter calculates $c_v := C - c_\phi$ in Step 3.
- For $i = \phi$, we have $a_\phi = g^{s_\phi}/\alpha'^{c_\phi}$ and $b_\phi = h^{s_\phi}/(\beta'/g^\phi)^{c_\phi}$ from Step 1.
- For $i = v$, we must check if $a_v = g^{s_v}/\alpha'^{c_v}{=}g^{w+rc_v}/\alpha'^{c_v}$ which holds since $a_v = g^w$ (from Step 1) and $\alpha^{c_v} = g^{rc_v}$. Similarly: $b_v = h^{s_v}/(\beta'/g^v)^{c_v} = h^{w+rc_v}/(\beta'/g^v)^{c_v}$ holds since $b_v = h^w$ and $\beta'/g^v = h^r$.

For the second property, we need to show that $C_0 + C_1 = H(A_0, B_0, A_1, B_1)$ and that given that $a_i = g^{s_i}/\alpha'^{c_i}$ and $b_i = h^{s_i}/(\beta'/g^i)^{c_i}$ hold (since the copier has access to a valid vote) it also holds that: $A_i = g^{S_i}/\alpha'^{C_i}$ and $B_i = h^{S_i}/(\beta'/g^i)^{C_i}$. This is straightforward by substituting the blinded variables A_i, B_i, C_i, S_i with their definitions. □

Lemma 14. *Protocol 5 is 2-special sound.*

Proof. Suppose a voter can (given the same first message $(\tilde{\alpha}, \tilde{\beta}, \pi), (a_0, b_0, a_1, b_1)$, and given that π verifies) provide answers to two different challenges C, C'. This means that for the two answers (c_0, s_0, c_1, s_1) and (c'_0, s'_0, c'_1, s'_1), we must have $c_i \neq c'_i$ for at least one $i \in \{0, 1\}$. We will now show that such a voter can calculate a witness for the vote's validity (i.e. the encryption randomness used in encrypting the vote).:

$$a_i = g^{s_i}/\alpha_i'^{c} \text{ and } a_i = g^{s'_i}/\alpha'^{c'_i} \text{ we have:}$$
$$g^{s_i}/\alpha'^{c_i} = g^{s'_i}/\alpha'^{c'_i}$$
$$g^{s_i - s'_i} = \alpha'^{c_i - c'_i} \text{ thus:}$$
$$\log_g \alpha' = \frac{c_i - c'_i}{s_i - s'_i}.$$

This gives us r' the logarithm of a'. To obtain the logarithm of α, one needs to subtract $\tilde{r}$, the logarithm of $\tilde{\alpha}$. This can be obtained in the same fashion from the proof on knowledge π. □

Lemma 15. *Given the view of the original voter, the blinded proof of knowledge* $((A_0, B_0, A_1, B_1), (C_0, S_0, C_1, S_1))$ *is unconditionally indistinguishable from a valid proof produced independently.*

Proof. We observe that $C_i = c_i + \Delta_i$ and $S_i = s_i + k_i$ with Δ_i and k_i being uniformly random in $\mathbb{Z}_p$. Thus, the challenges and responses are independent of the ones used in the original proof. For the commitments, we note that given (C_0, S_0, C_1, S_1), the values of (A_0, B_0, A_1, B_1) are uniquely determined (because for any valid proof: $A_i = g^{S_i}/\alpha'^{C_i}$ and $B_i = h^{S_i}/(\beta'/g^i)^{C_i}$), so if a voter is able to distinguish $(A_0, B_0, A_1, B_1), (C_0, S_0, C_1, S_1)$ from an independent proof he would also be able to distinguish (C_0, S_0, C_1, S_1) which we showed to be distributed identically. □

Our goal is to ensure that the blinded protocol does not leak the value of the voter's vote to one of the other parties.

Lemma 16. *Protocol 5 achieves computational zero-knowledge in the random oracle model, under the DDH assumption.*

Proof. We will describe a simulator. Intuitively, the simulator will program the random oracle to fake π so that it can switch to a fresh α', β' that is independent of α, β. It will then follow the same strategy as an honest prover.

The simulator first samples $r' \leftarrow \mathbb{Z}_p$, $v \leftarrow \{0,1\}$. It then sets $(\alpha', \beta') = (g^{r'}, h^{r'})$, and $\tilde{\alpha} = \alpha'/\alpha$, $\tilde{\beta} = \beta'/\beta$. It then produces a proof π_s that $\tilde{\alpha}, \tilde{\beta}$ share discrete logs as follows: it selects a challenge e_π and answer s_π and programs the random oracle so that $H(g^{s_\pi}/\tilde{\alpha}'^{e_\pi}, h^{s_\pi}/\tilde{\beta}^{e_\pi}) = e_\pi$. Then, $\pi = ((g^{s_\pi}/\tilde{\alpha}^{e_\pi}, h^{s_\pi}/\tilde{\beta}^{e_\pi}), s_\pi)$.

From this point on, the simulator follows the same protocol as the prover: $w \leftarrow \mathbb{Z}_p$ and let $a_v := g^w, b_v := h^w$. Let $\phi := 1 - v$ and $c_\phi, s_\phi \leftarrow \mathbb{Z}_p$ and let $a_\phi := g^{s_\phi}/\alpha'^{c_\phi}$ and $b_\phi := h^{s_\phi}/(\beta'/g^\phi)^{c_\phi}$. Then, it sends $(\tilde{\alpha}, \tilde{\beta}, \pi), (a_0, b_0, a_1, b_1)$ to its copier oracle to receive a challenge, and replies as the standard prover would.

We now compare the distributions of real and simulated transcripts. First we point out that if the simulator correctly guesses v to have the same value as the plaintext of α, β, the transcript is distributed identically to a real one. Thus, an efficient distinguisher between real and simulated transcripts would also distinguish between simulated transcripts with "correct" and "incorrect" guesses. This directly breaks the IND-CPA security of ElGamal. □

We can also use the above simulator to prove that the protocol is also zero-knowledge with respect to the final prover, Helios. Since the Fiat-Shamir heuristic [FS87] is used to replace the verifier's challenge with a hash of the the prover's commitments, Helios is unable to deviate from honest behaviour.

We also point out that the simulator may also be run with the value of v given as an input, without impacting zero knowledge. This is key to reconciling copying and privacy as it enables us to provide a copying oracle to the adversary: if the adversary is seeing the "real" board we allow him to copy as normal but if he is seeing the "fake" one, we use the simulator with the vote from the "real" board.

8.5.3 A Combined Protocol for Blinded Copying

The vote blinding transformation of Sect. 8.4 and the proof blinding protocol (Protocol 4) can each partially blind a ballot (the vote and the proof respectively). They can be easily combined to completely blind a ballot as follows: The copier executes the proof blinding protocol with the cooperation of the voter but does not submit the resulting ballot V. Instead, he proceeds to apply the vote blinding transformation to V, producing V' which he then submits to Helios. If Helios switches to the strong Fiat-Shamir transformation, the two protocols need to be combined

Theorem 16. *The combined ballot copying protocol is complete, sound and zero-knowledge in the random oracle model. Furthermore, assuming the DDH assumption holds, the ballots produced are accepted by Helios and indistinguishable from random valid ballots, even for the voter.*

Proof. Completeness, soundness and honest-verifier zero-knowledge under the random oracle model are satisfied by the proof copying protocol and are not impacted by the transformation (Lemma 12). Indistinguishability holds because of Lemmata 11 and 15. □

8.6 Further Work

We have described a protocol which enables voters to allow people who trust them to copy their vote without revealing it in the process. This can be used as an alternative to public endorsements. In settings where one person's expertise or judgement is well regarded our protocol offers the ability for others to trust his judgement without forcing him to reveal his opinion –this can be especially important in small, local elections where revealing one's vote can lead to rivalries (of course in a small or close election the tally [DK05] or even the result might reveal information).

Blinded vote copying would also reduce the power of traditional voting blocs. A voting bloc is a club or special interest group that coordinates its voting. They achieve stronger [Pen46] representation compared to individual voters by not diffusing their votes. The trust requirements for blinded vote copying are more relaxed than in a typical voting bloc since the "leader" does not need to make his vote public. By making the creation of voting blocs easier we thus create a more even voting field without needing to change the electoral system.

It would also be interesting to replace the original voter with a coalition of voters, essentially providing a framework (thus avoiding the complexity of secure multi-party computation) for holding a primary election amongst the members of the coalition. This can lower the barrier for creating a voting bloc further since there is no need for a single person to be singled out as the decision maker.

Finally, a voting system built with copying in mind may be able to provide users with a stronger mix of functionality and security. The high risk of coercion present in Helios might be too high a price to pay for vote copying. Building support for copying into the protocol may allow us to obtain similar functionality without relying on receipts.

Chapter 9

BeleniosRF

Returning to the issue of receipt freeness, we now offer a more orthodox approach. We propose a new voting scheme, BeleniosRF, that offers both receipt-freeness and verifiability. It is receipt-free in a strong sense, meaning that even dishonest voters cannot prove how they voted. Furthermore, we achieve receipt freeness without burdening the user with any additional effort (as in the case of revoting as an anti-coercion strategy).

We provide a game-based definition of receipt-freeness for voting protocols with non-interactive ballot casting, which we name *strong receipt-freeness* (sRF). To our knowledge, sRF is the first game-based definition of receipt-freeness in the literature, and it has the merit of being particularly concise and simple. Built upon the Helios protocol, BeleniosRF inherits its simplicity and does not require any anti-coercion strategy from the voters. We implement BeleniosRF and show its feasibility on a number of platforms, including desktop computers and smartphones.

The research presented in this chapter is joint work with Véronique Cortier, Georg Fuchsbauer and David Galindo, and was published in [CCFG16]. The author contributed to the investigation mainly in designing the voting protocol and SRC scheme in collaboration with Georg Fuchsbauer. The author also contributed the implementation and the majority of the benchmarking.

9.1 Introduction

Electronic voting protocols should achieve two antagonistic security goals: privacy and verifiability. Additionally, they must be practical, from a usability, operational, and efficiency point of view. Privacy can be expressed via several, increasingly demanding security properties.

- Basic *ballot privacy* guarantees that no one can learn how a voter voted.
- *Receipt-freeness* ensures that a voter cannot prove to anyone how she voted. While privacy protects honest voters, receipt-freeness aims at protecting vote privacy even when voters willingly (or unwillingly) interact with an attacker.
- *Coercion-resistance* should allow an honest voter to cast her vote even if she is, during some time, fully under the control of an attacker. Coercion-resistance typically requires revoting.

Conversely, verifiability ensures that voters' ballots are included in the ballot box (*individual verifiability*), that the result corresponds to the content of the ballot box (*universal verifiability*) and that ballots come only from voters entitled to vote (*eligibility verifiability*).

Helios [Adi08, AdMPQ09] is a scheme that "only" achieves privacy and verifiability and is based on a voting system by Cramer, Gennaro and Schoenmakers [CGS97b] with modifications proposed by Benaloh [Ben07]. As emphasised by its authors, Helios should only be used in low-coercion environments. Indeed, a voter may easily reveal how she voted by exhibiting the randomness used by her device to compute her ballot (Sect. 8.2.1); one can then re-encrypt the claimed vote and check if the encryption is contained in the public bulletin board. Helios is thus not receipt-free if we assume voters have a reasonable amount of control of their devices.

To our knowledge, Civitas [JCJ05, CCM08] is the only scheme that achieves both verifiability and coercion-resistance, without requiring a great deal of interaction between the ballot box or the election authorities and the voter (such as [BT94, CLW08]). While the scheme is a foundational work, it seems difficult to use it in large-scale elections mainly for two reasons. First, its coercion resistance design requires that voters must be allowed to cast multiple ballots, with checks against duplicates taking place during tallying. This involves each cast ballot being checked against each other, requiring $O(n^2)$ operations –where n is the number of received ballots, which opens the way to denial-of-service attacks. Second, to achieve coercion-resistance, a voter should be able to adopt an *anti-coercion strategy* (in Civitas, a voter has to lie about her true credential) and then later *revote* for her true choice once she is freed from the attacker. We believe that this scenario is unrealistic in many cases, as it requires cryptographic skills and a heavy infrastructure to realise an untappable channel (e.g. in-person registration).

It is also worth noticing that in most countries revoting is not allowed, as for example in Australia, France, Spain, Switzerland and the United Kingdom. The only exceptions we are aware of are Estonia and the Internet voting pilots for the parliamentary elections in 2011 and 2013 in Norway. Indeed, casting a vote is usually seen as an official act that a citizen should perform with care and revoting is perceived as "changing one's mind". Most election authorities are strongly opposed to this practice. While this way of thinking might be a cultural aspect inherited from traditional paper ballot systems, it is foreseeable that it will take time before countries change their electoral rules in order to adopt a revote policy.

9.1.1 Our Contributions

Building upon a recent variant of Helios, called Belenios [CGGI14, GCG15], and a cryptographic primitive called *signatures on randomizable ciphertexts* [BFPV11a], we propose a receipt-free version of Helios, which we call BeleniosRF. In our scheme a voter cannot prove how she voted, even if she is provided with all the ballot material by the coercer. Interestingly, our scheme does not demand any strategy of the voter; in particular, it does not require the active participation of a voter to deceive a coercer that is asking for a receipt. For example, a voter does not need to lie or produce fake credentials as in Civitas, she simply *has no way* to prove how she voted. This represents a huge improvement in usability from the voter's point of view: all that is required of the voter is to vote.

We show that our scheme BeleniosRF is receipt-free in a strong sense, meaning that even a dishonest voter using a voting client that has been tampered with cannot prove how she voted. We formalise this property, called *strong receipt-freeness* (sRF), via a game-based definition building on the privacy definition recently proposed by Bernhard *et al.* [BCG$^+$15]. We view this formal definition of receipt-freeness, which applies to non-interactive ballot casting protocols, as the first contribution of this work. We call it *strong* receipt-freeness to emphasise that in non-interactive protocols an attacker has less room to build a receipt. Indeed, in the absence of interaction the adversary does not obtain information from the voting server apart from what is displayed on the bulletin board; hence any receipt must be built by the adversary locally and before submitting the ballot.

We claim sRF is the first game-based receipt-freeness definition in the literature accounting for a voter that is corrupted during the voting phase. Additionally, sRF has the merit of being simple and concise, potentially allowing for simpler proofs. In doing so we give a new formulation for the receipt-freeness definition by Benaloh and Tuinstra [BT94] and highlight that receipt-freeness can be achieved without asking the voters to vote several times and cancel previously submitted ballots, and without requiring an untappable channel. All we need to assume is that the attacker is not permanently eavesdropping the communication between the voting server and the voter, an assumption made by all previous constructions of receipt-free or coercion-resistant voting schemes.

A key ingredient of BeleniosRF is a *randomization service*, a role that we assume is played by the voting server (also called ballot box), but which could be played by a different server. The randomization service is in charge of *re-randomizing* the ballot cast by a voter. BeleniosRF's receipt-freeness then relies on the fact that the randomness contained in the ballot displayed in the bulletin board *is not* under the control of the voter. Both the voter and the randomization service contribute to the randomness that is implicit in the voter's ballot as displayed on the bulletin board. In fact, in light of the impossibility result of [CFP$^+$10], the existence of a randomization agent is assumed in most constructions that claim to be receipt-free. Here however, we do not rely on letting voters vote multiple times or on the existence of a trusted token for each voter (such as e.g. [Oka97, HS00, CLW08, Hir10]).

The foremost challenge in achieving receipt-freeness non-interactively and via a randomization service is to prevent the latter from changing the voter's intent when re-randomizing. The only existing non-interactive proposal [BFPV11a] claiming receipt-freeness uses a powerful cryptographic primitive called *signatures on randomizable ciphertexts*. It consists of a signature scheme and a public-key encryption scheme that is randomizable (that is, given a ciphertext, anyone can create a fresh ciphertext of the same plaintext—without knowing it). The primitive provides an additional functionality: given a signature on a ciphertext, anyone can randomise the ciphertext and *adapt* the signature to the new ciphertext, that is, produce a signature that is valid on the new ciphertext—and all that knowing neither the decryption key nor the signing key nor the plaintext. On the other hand, unforgeability guarantees that it is infeasible to compute a signature on a ciphertext that encrypts a message of which no encryption has been signed.

Alas, Blazy *et al.* [BFPV11a] did not provide a receipt-freeness definition nor a proof. By exhibiting a ballot-copying attack adapted from [CS13], we demonstrate that their scheme is not receipt-free, worse, it is not even ballot-private. Our scheme fixes the Blazy *et al.* construction by binding the ciphertexts to voters, while still inheriting the randomizability from Groth-Sahai non-interactive proofs [GS08], upon which the previous proposal heavily relies, this provides the required non-malleability needed to thwart the attack.

We start with giving a new instantiation of signatures on randomizable ciphertexts, which we show yields an RCCA-secure public-key encryption scheme [CKN03], from which we build a non-interactive[1] receipt-free e-voting scheme as follows:

- As in Belenios, each voter has a personal signature key pair, in addition to authentication means to the ballot box (typically a login and password).
- Each voter encrypts and signs their ballot and includes a proof of knowledge to prevent ballot malleability.
- Upon receiving a ballot, the server re-randomises the ballot (including the signature and proof) and adapts the corresponding signature and proof before publishing it.
- While only the re-randomised version is posted on the bulleting board, we can still check the signature and proof as are part of the re-randomisation.

Receipt-freeness comes from the fact that a voter no longer has control over, nor knowledge of, the randomness used to form the final ballot stored in the ballot box. On the other hand, even after the voting server re-randomises the ballot cast by the voter's voting device, the voter can still verify that her legitimate ballot is present, as the re-randomised ciphertext comes with a signature that is valid under the voter's verification key. By unforgeability of the signature primitive, the vote cannot have been altered by the ballot box, which we show implies verifiability.

Our final contribution consists of assessing the feasibility of BeleniosRF; for this purpose we implemented and measured the efficiency of a Javascript voting client (see Section 9.7).

9.2 Comparison to Related Work

Our definition requires that an adversary cannot distinguish whether a voter votes for either a or b, even if the attacker provides the voter in advance with all the cryptographic material

[1]After successful authentication between the voter and the ballot box, ballot casting is non-interactive.

(such as randomness to be used to cast the ballot). Interestingly, this definition does not require the voter to follow a "strategy" to fool the coercer.

BeleniosRF has one disadvantage compared to Helios and Belenios: it requires that voters generate keys by themselves and have them endorsed by a registrar. Helios does not use signatures on ballots, whereas Belenios opts to have the keys be generated centrally and distributed to voters.

What makes BeleniosRF different, is the rerandomization of ballots, which is used to guarantee receipt-freeness. If the registrar knows a users private signing key, and it colludes with the voting server collude, they could undetectably change a voter's choice. This is because (rerandomized) signatures on different ciphertexts encrypting the same message cannot be linked. Thus, the registrar could sign a malicious ballot using a voter's key, and the voting server would pretend that the malicious ballot is a rerandomization of the one cast. Helios and Belenios are immune to this, since they do not use rerandomization (or even ballot signing, in the case of Helios).

This solution differs from the Belenios approach, where voters receive their signing keys from the registrar for the sake of usability. This is just another manifestation of the usual tension between usability, privacy and verifiability in e-voting systems (and computer security systems in general), in the sense that increasing one of them entails a decrease of at least one of the others. One may opt to instead use an *issuing* registrar (as opposed to an *endorsing* one) may still be used, but requires that either the voting server or the registrar remain honest (cf. Section 9.6.3).

9.3 Receipt-Freeness

We now formally define receipt-freeness and start by providing the syntax of a voting system, inspired by [CGGI14, BCG$^+$15].

9.3.1 Syntax of a Voting System

Election systems typically involve several entities. For the sake of simplicity we consider each entity to consist of only one individual but note that all of them could be thresholdised.

1. *Election administrator*: denoted by $\mathcal{E}$, is responsible for setting up the election; it publishes the identities id of eligible voters, the list of candidates and the result function ρ of the election (typically counting the number of votes received by every candidate).

2. *Registrar*: denoted by $\mathcal{R}$, is responsible for registering the public credentials of users

after verifying the users control the corresponding secret ones .

3. *Trustee:* denoted by $\mathcal{T}$, is in charge of tallying and publishing a final result.

4. *Voters*: the eligible voters are denoted by $id_1, \ldots, id_\tau$.

5. *Ballot-box (voting server) manager*: denoted by $\mathcal{B}$, is responsible for processing and storing valid ballots in the ballot box BB, and for publishing PBB, the public view of BB, also called (public) bulletin board.

The following syntax considers *single-pass* schemes, that is, systems where voters only have to post a single message to the board, i.e. ballot casting is non-interactive. A voting protocol $\mathcal{V} = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Valid}, \mathsf{Append}, \mathsf{Publish}, \mathsf{VerifyVote}, \mathsf{Tally}, \mathsf{Verify})$ is relative to a family of result functions $\{\rho_\tau\}_{\tau \geq 1}$ for $\tau \in \mathbb{N}$, with $\rho_\tau \colon \mathbb{V}^\tau \to \mathbb{R}$, where $\mathbb{V}$ is the set of admissible votes and $\mathbb{R}$ is the result space.

$\mathsf{Setup}(1^\lambda)$, on input a security parameter 1^λ, outputs an election public/secret key pair $(\mathbf{pk}, \mathbf{sk})$, where $\mathbf{pk}$ could contain a list of credentials L. We let $\mathbf{pk}$ be an implicit input of the remaining algorithms.

$\mathsf{Register}(id)$, is an interactive protocol between the registrar and a user with identifier id. The user creates usk_{id} and its public credential upk_{id}, and forwards upk_{id} to the registrar along with a proof of knowledge of the corresponding secret key, π_{id}. The registrar, after checking, π_{id} adds it to the list $L = \{upk_{id}\}$.

$\mathsf{Vote}(id, upk, usk, v)$ is run by voter id with credentials upk, usk to cast her vote $v \in \mathbb{V}$. It outputs a ballot b, which is sent to the voting server (possibly through an authenticated channel).

$\mathsf{Valid}(\mathsf{BB}, b)$ takes as input the ballot box BB and a ballot b and checks the validity of the latter. It returns $\top$ for valid ballots and $\bot$ for invalid ones (e.g. ill-formed, containing duplicated ciphertext from the ballot box…).

$\mathsf{Append}(\mathsf{BB}, b)$ updates BB with the ballot b. Typically, this consists in adding b as a new entry to BB, but more involved actions might be possible (as in our scheme).

$\mathsf{Publish}(\mathsf{BB})$ outputs the public view PBB of BB. Often one simply has $\mathsf{Publish}(\mathsf{BB}) = \mathsf{BB}$.

$\mathsf{VerifyVote}(\mathsf{PBB}, id, upk, usk, b)$ is run by voters for checking that their ballots will be included in the tally. On inputs the public board PBB, a ballot b, and the voter's identity and credentials id, usk, upk, it returns $\top$ or $\bot$.

$\mathsf{Tally}(\mathsf{BB},\mathbf{sk})$ on inputs the ballot box BB and the secret key $\mathbf{sk}$, outputs the tally r and a proof of correct tabulation Π. If the election is declared invalid then $r := \perp$.

$\mathsf{Verify}(\mathsf{PBB},r,\Pi)$, on inputs the public bulletin board PBB and (r,Π), checks whether Π is a valid proof of correct tallying for r. If so, it returns $\top$, and $\perp$ otherwise.

The exact implementation of these algorithms depends on the concrete voting protocol. In particular, the notion of public and private credentials of a voter varies a lot. For example upk_{id} might be simply the identity of the voter or may correspond to her signature-verification key.

9.3.2 Strong Receipt-Freeness

Intuitively, privacy ensures that an adversary cannot learn the vote of an honest voter. Receipt-freeness furthermore guarantees that a voter cannot prove how she voted, even if she willingly provides information to, or follows instructions by, the adversary. This captures the seminal intuition from Benaloh and Tuinstra [BT94]. The latter insisted that a reasonably private electronic voting protocol should emulate traditional voting in a voting booth: it should *allow* voters to conceal their individual votes and, at the same time, *prevent* them from revealing their vote. Voters should not be able to give away the privacy of their vote granted by the voting protocol, even if they are willing to.

Building upon a definition of privacy recently introduced [BCG$^+$15], we argue that this requirement can be formalised for single-pass schemes by simply providing the adversary with an additional oracle $\mathcal{O}\mathsf{receiptLR}$, which allows him to submit his own ballots on behalf of a dishonest voter. Apart from immediately implying ballot privacy, this simple formalization captures several important scenarios:

- A voter who wants to convince a vote buyer of how she voted may prepare her ballot in an arbitrary way that allows him to construct a convincing receipt (e.g., consider a voter that uses biased random coins to build her ballot and to prove how she voted [GGR09]).
- A voter that might have been corrupted before the ballot casting phase may just follow the instructions given to her by the adversary (as in [JCJ05]).
- A voter can record, but also forge, its interaction with the ballot box (as in [BT94]).

As in previous formal or intuitive definitions of receipt-freeness, we assume the adversary is not monitoring the interaction between the voter and the voting server. However, the voter can record this interaction, and later on present this information (or any transformation thereof) to the adversary.

Formally, we consider two games, Game 0 and Game 1, defined by the oracles in Figure 9.1. In both games BB_0 and BB_1 are ballot boxes that start out empty. Box BB_0 corresponds to the real election (that will be tallied) and BB_1 is a fake ballot box which the adversary's task is to distinguish from BB_0. In Game β the adversary has indirect access to BB_β, that is, she can see the public part of that box at any time. The game $\mathsf{Exp}^{\mathsf{srf},\beta}_{\mathcal{A},\mathcal{V}}$ provides an adversary $\mathcal{A}$ access to the oracles defined in Figure 9.1, which intuitively proceed as follows:

$\mathcal{O}\mathsf{init}$ generates secret and public keys for the election; the public key is returned to the adversary. If $\beta = 1$, it also returns auxiliary information *aux* to be used by a simulator $\mathsf{SimProof}$ introduced below.

$\mathcal{O}\mathsf{reg}$, on input an identifier *id*, initialises *id*'s credentials (upk, usk) by running $\mathsf{Register}(id)$. It gives *upk* to the adversary.

$\mathcal{O}\mathsf{corrupt}$ is used by the attacker to obtain the credentials (upk, usk) of a registered voter.

$\mathcal{O}\mathsf{voteLR}$, a left-or-right oracle, takes two potential votes (v_0, v_1) for an honest user *id*, produces ballots b_0 and b_1 for these votes and places them in the ballot boxes (one in BB_0 and one in BB_1), provided that $v_0, v_1 \in \mathbb{V}$.

$\mathcal{O}\mathsf{cast}$ allows the adversary to cast a ballot b on behalf of any party. If the ballot is valid with respect to BB_β, it is placed in both ballot boxes.

$\mathcal{O}\mathsf{receiptLR}$ allows an adversarial voter *id* to cast a ballot b_1 in BB_1 and a ballot b_0 in BB_0. If each ballot b_0, b_1 is valid with respect to its respective ballot box, then the ballots are appended by running $\mathsf{Append}(\mathsf{BB}_0, b_0)$ and $\mathsf{Append}(\mathsf{BB}_1, b_1)$. This allows the adversary to encode special instructions in the ballots that could later serve as the basis for a vote receipt (e.g. as in [GGR09]).

$\mathcal{O}\mathsf{board}$ models the adversary's ability to see the publishable part of the board. It returns $\mathsf{Publish}(\mathsf{BB}_\beta)$.

$\mathcal{O}\mathsf{tally}$ allows the adversary to see the result of the election. In both games the result is obtained by tallying a valid BB_0; the proof of correct tabulation is however simulated in the second world, i.e., for $\beta = 1$.

```
$\mathcal{O}\mathsf{init}$
    for $\beta = 0$
        $(\mathbf{pk}, \mathbf{sk}) \leftarrow \mathsf{Setup}(1^k)$
        return $\mathbf{pk}$
    for $\beta = 1$
        $(\mathbf{pk}, \mathbf{sk}, aux) \leftarrow \mathsf{SimSetup}(1^k)$
        return $\mathbf{pk}$

$\mathcal{O}\mathsf{reg}(id)$
    If $id$ was not previously
    queried, then run $\mathsf{Register}(id)$
    and set
    $\mathcal{U} := \mathcal{U} \cup \{(id, upk_{id}, usk_{id})\}$
    return $upk_{id}$.

$\mathcal{O}\mathsf{corruptU}(id)$
    On a registered voter $id$, output
    $(upk_{id}, usk_{id})$ and set
    $\mathcal{CU} := \mathcal{CU} \cup \{(id, upk_{id})\}$.

$\mathcal{O}\mathsf{cast}(id, b)$
    If $\mathsf{Valid}(\mathsf{BB}_\beta, b) = \perp$ then
    return $\perp$. Else
    $\mathsf{Append}(\mathsf{BB}_0, b)$
    $\mathsf{Append}(\mathsf{BB}_1, b)$.

$\mathcal{O}\mathsf{voteLR}(id, v_0, v_1)$
    If $v_0 \notin \mathbb{V}$ or $v_1 \notin \mathbb{V}$ then
    return $\perp$. Otherwise:
    $b_0 := \mathsf{Vote}(id, upk_{id}, usk_{id}, v_0)$
    $b_1 := \mathsf{Vote}(id, upk_{id}, usk_{id}, v_1)$
    $\mathsf{Append}(\mathsf{BB}_0, b_0)$;
    $\mathsf{Append}(\mathsf{BB}_1, b_1)$

$\mathcal{O}\mathsf{receiptLR}(id, b_0, b_1)$
    If $id \notin \mathcal{CU}$ return $\perp$.
    If $\mathsf{Valid}(\mathsf{BB}_0, b_0) = \perp$ or
    $\mathsf{Valid}(\mathsf{BB}_1, b_1) = \perp$
    return $\perp$. Else
    $\mathsf{Append}(\mathsf{BB}_0, b_0)$
    $\mathsf{Append}(\mathsf{BB}_1, b_1)$

$\mathcal{O}\mathsf{board}()$
    Return $\mathsf{Publish}(\mathsf{BB}_\beta)$

$\mathcal{O}\mathsf{tally}()$ for $\beta = 0$
    $(r, \Pi) \leftarrow \mathsf{Tally}(\mathsf{BB}_0, \mathbf{sk})$
    return $(r, \Pi)$

$\mathcal{O}\mathsf{tally}()$ for $\beta = 1$
    $(r, \Pi) \leftarrow \mathsf{Tally}(\mathsf{BB}_0, \mathbf{sk})$
    $\Pi' \leftarrow \mathsf{SimProof}_{aux}(\mathsf{BB}_1, r)$
    return $(r, \Pi')$
```

Figure 9.1: Oracles defining experiments $\mathsf{Exp}^{\mathsf{srf},\beta}_{\mathcal{A},\mathcal{V}}(\lambda)$ for $\beta = 0, 1$. The games differ in the way the tallying oracle creates auxiliary data, in the board displayed to the adversary in response to $\mathcal{O}\mathsf{board}$ queries, and the board against which ballots are validated.

We demand that the adversary first calls $\mathcal{O}\mathsf{init}$, then oracles $\mathcal{O}\mathsf{reg}, \mathcal{O}\mathsf{corruptU}, \mathcal{O}\mathsf{voteLR}$, $\mathcal{O}\mathsf{cast}, \mathcal{O}\mathsf{receiptLR}, \mathcal{O}\mathsf{board}$ in any order, and any number of times. Finally, $\mathcal{A}$ can call $\mathcal{O}\mathsf{tally}$; after it receives its reply, $\mathcal{A}$ must return a guess of the bit β. The guess bit is the result returned by the game.

Inherited from ballot privacy [BCG+15], Definition 30 uses simulators SimSetup and SimProof to model the fact that the proof should not reveal anything, as it is "zero-knowledge".

Definition 30 (sRF). *Let* $\mathcal{V} = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Valid}, \mathsf{Append}, \mathsf{VerifyVote}, \mathsf{Publish}, \mathsf{Tally}, \mathsf{Verify})$ *be a voting protocol for a set* ID *of voter identities and a result function* ρ. *We say that* $\mathcal{V}$ *has* strong receipt-freeness *if there exist algorithms* SimSetup *and* SimProof *such that no efficient adversary can distinguish between games* $\mathsf{Exp}^{\mathsf{srf},0}_{\mathcal{B},\mathcal{V}}(\lambda)$ *and* $\mathsf{Exp}^{\mathsf{srf},1}_{\mathcal{B},\mathcal{V}}(\lambda)$ *defined by the oracles in Figure 9.1; that is, for any efficient algorithm* $\mathcal{A}$ *the following is negligible in* λ*:*

$$\left| \Pr\left[\mathsf{Exp}^{\mathsf{srf},0}_{\mathcal{A},\mathcal{V}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{srf},1}_{\mathcal{A},\mathcal{V}}(\lambda) = 1\right] \right| .$$

In protocols with non-interactive ballot casting an adversary does not receive any output from its interaction with the ballot box (apart from the public view of the protocol run), the sRF adversary must therefore build a receipt using local data only, and before casting the ballot. An adversary might encode arbitrary instructions in b_β, for instance making those instructions dependent on the vote v_β; e.g. he could set the least significant bit of b_β equal to $v_\beta \in \{0,1\}$. Intuitively, *strong receipt-freeness* implies that a ballot b_0 could be replaced by a ballot b_1, both submitted via the oracle $\mathcal{O}\mathsf{receiptLR}$, without the adversary noticing. Thus a receipt, i.e. a proof for a certain vote having been cast, cannot exist as $\mathcal{O}\mathsf{receiptLR}$ captures all what a RF adversary can do.

This definition does not assume that the voter is capable of successfully applying some anti-coercion strategy (in contrast to [MN06]). We believe this to be important in practice for two reasons. First, this is of course much easier to use: with our definition, the system *is* receipt-free by construction and there is no need to instruct voters how they should proceed to lie about their vote. Second, we need not assume that revoting is allowed (our definition accommodates any revoting policy though, including no revote). This is important since most countries forbid revoting.

As expected, strong receipt-freeness trivially implies BPRIV privacy [BCG+15], since BPRIV equals sRF except that there is no oracle $\mathcal{O}\mathsf{receiptLR}$.

Helios. Under the RF definition provided in [KZZ15] the Helios protocol would be receipt-free. In contrast, under our definition Helios *is not* receipt-free. Indeed, if the adversary is allowed to cast different ballots b_0, b_1 to the ballot boxes $\mathsf{BB}_0, \mathsf{BB}_1$, respectively, then distinguishing Game 0 from Game 1 is trivial. This is due to the fact that in Helios PBB contains the encryption of the votes, so it suffices for an adversary to produce different encryptions c, d and check which one is showing up when calling oracle $\mathcal{O}\mathsf{board}$.

Interestingly, we believe that a Helios instantiation in which voting devices are built upon trusted hardware tokens that conceal the randomness used for encryption (as proposed by Magkos *et al.* [MBC01]) satisfies sRF, when interpreting trusted tokens as preventing $\mathcal{A}$ from accessing the $\mathcal{O}\mathsf{receiptLR}$ oracle—in which case sRF collapses to ballot privacy. This shows the flexibility of our definition. Moreover, we are confident that the same result applies to [KZZ15].

9.4 Building Blocks

Before describing our voting scheme, we first present the necessary cryptographic building blocks.

9.4.1 Assumptions and Primitives

We will work in asymmetric bilinear groups and assume the existence of a *bilinear-group generator* GrpGen, which on input 1^λ outputs $(p, \mathbb{G}_1, \mathbb{G}_2, g_1, g_2, \mathbb{G}_T, e)$, where p is a prime of length λ, $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of order p, g_1 is a generator of $\mathbb{G}_1$, g_2 is a generator of $\mathbb{G}_2$, and e is a bilinear map $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ such that $e(g_1, g_2)$ generates $\mathbb{G}_T$. The following was discussed in [BLS04, p. 304] and defined in [BFPV11a].

Definition 31 (CDH$^+$)**.** *The CDH$^+$ assumption holds for* GrpGen *if for* $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, g_1, g_2, \mathbb{G}_T, e) \leftarrow \mathsf{GrpGen}(1^\lambda)$*, and for* $a, b \leftarrow \mathbb{Z}_p$*, for every p.p.t. adversary given* $(\mathcal{G}, g_1^a, g_2^a, g_1^b)$*, the probability that it outputs* g_1^{ab} *is negligible in* λ*.*

The next assumption implies the security of ElGamal encryption in both groups $\mathbb{G}_1$ and $\mathbb{G}_2$:

Definition 32 (SXDH)**.** *The Symmetric external Diffie-Hellman assumption (SXDH) holds for* GrpGen *if for* $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, g_1, g_2, \mathbb{G}_T, e) \leftarrow \mathsf{GrpGen}(1^\lambda)$*,* $a, b, c \leftarrow \mathbb{Z}_p$ *and for both* $i \in \{1, 2\}$*, p.p.t. adversaries only distinguish* $(\mathcal{G}, g_i^a, g_i^b, g_i^{ab})$ *from* $(\mathcal{G}, g_i^a, g_i^b, g_i^c)$ *with advantage negligible in* λ*.*

Variant ElGamal Encryption. In this chapter we will use a variation of the ElGamal encryption scheme from Section 3.3.5. This is to improve efficiency in the proofs that follow. The difference is that the message now lies in the component containing the public key instead of the one containing the generator.

We will define encryption for messages in $\mathbb{G}_1$ from an asymmetric bilinear group $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, g_1, g_2, \mathbb{G}_T, e)$ and show that it is randomizable.

$\mathsf{KeyGen}(\mathcal{G}, i)$: Choose $d \leftarrow \mathbb{Z}_p$ and define $P := g_1^d$. Return $(pk = P, dk = d)$.

$\mathsf{Encrypt}(P, M; r)$: Using randomness $r \in \mathbb{Z}_p$, output $c = (c_1 := g_1^r, c_2 := M \cdot P^r)$.

$\mathsf{Decrypt}(d, c = (c_1, c_2))$: Output $M := c_2 \cdot c_1^{-d}$.

$\mathsf{Random}(P, c = (c_1, c_2); r')$: Using randomness $r' \in \mathbb{Z}_p$, output $c' := (c_1 \cdot g_1^{r'}, c_2 \cdot P^{r'})$.

This scheme is IND-CPA secure assuming hardness of DDH in $\mathbb{G}_1$, which follows from SXDH. It is perfectly randomizable as $\mathsf{Random}(pk, \mathsf{Encrypt}(pk, M; r); r') = \mathsf{Encrypt}(pk, M; r + r')$.

Groth-Sahai Proofs. Groth-Sahai (GS) proofs [GS08] allow us to prove satisfiability of equations involving group elements from $\mathbb{G}_1$ or $\mathbb{G}_2$ and scalars. We will use them to prove consistency and knowledge of encryptions. On input a bilinear group $\mathcal{G}$, $\mathsf{Setup}_{\mathrm{GS}}$ outputs a common reference string (CRS) $crs \in \mathbb{G}_1^4 \times \mathbb{G}_2^4$. The CRS is used to commit to group elements $X \in \mathbb{G}_1$, which we denote by $\mathcal{C}_1(X)$, and elements $Y \in \mathbb{G}_2$, denoted by $\mathcal{C}_2(Y)$. Moreover, $\mathcal{C}'_i(x)$ denotes a commitment to a scalar, which can be made in $\mathbb{G}_1$ $(i = 1)$ and $\mathbb{G}_2$ $(i = 2)$. The GS system lets us prove that committed values satisfy certain equations.

Under a CRS computed via $\mathsf{Setup}_{\mathrm{GS}}$, commitments are perfectly binding and the proofs are perfectly sound. That is, the values uniquely determined by the commitments satisfy the proved equation. Moreover, the committed values can be extracted using an extraction trapdoor ξ that can be computed together with the CRS. We denote this by $(crs, \xi) \leftarrow \mathsf{Setup}_{\mathrm{GS}}^{(x)}(\mathcal{G})$.

There is an alternative CRS-generation algorithm $\mathsf{Setup}_{\mathrm{GS}}^{(h)}$, which outputs (crs', td). Commitments made under crs' contain no information about the committed value and the trapdoor td allows simulation of proofs. As CRSs output by $\mathsf{Setup}_{\mathrm{GS}}$ and $\mathsf{Setup}_{\mathrm{GS}}^{(h)}$ are indistinguishable under SXDH, GS proofs are computationally zero-knowledge. Moreover, GS proofs are randomizable [FP09], that is, given commitments and proofs, one can (without knowing the witness) create a fresh set of commitments and proofs.

9.4.2 Signatures on Randomizable Ciphertexts

The primitive introduced by Blazy *et al.* [BFPV11a] consists of the following algorithms: Setup, on input the security parameter 1^λ, outputs the parameters (such as the bilinear group); $\mathsf{SKeyGen}$ outputs a pair of signing key and verification key (sk, vk), $\mathsf{EKeyGen}$ outputs a pair of encryption and decryption key (pk, dk). $\mathsf{SKeyGen}$ together with Sign and Verify constitutes a signature scheme and $\mathsf{EKeyGen}$ with $\mathsf{Encrypt}$ and $\mathsf{Decrypt}$ a public-key encryption scheme.

As the signature and the encryption scheme are used together, these algorithms have extensions Sign^+ and Verify^+, which additionally take the encryption key pk as input; and $\mathsf{Encrypt}^+$, $\mathsf{Decrypt}^+$, which also take the verification key vk.

Randomizability. The main feature of signatures on randomizable ciphertexts (SRC) is an algorithm Random^+, which takes pk, vk, a ciphertext c under pk and a signature σ on c valid under vk, and outputs a re-randomization c' of c together with a signature σ', valid on c'.

An output of Random^+ is distributed like a fresh encryption of the plaintext of c and a fresh signature on it; formally, for all messages m, $(pk, dk) \in [\mathsf{EKeyGen}(\mathcal{G})]$, $(vk, sk) \in [\mathsf{SKeyGen}(\mathcal{G})]$, $c \in [\mathsf{Encrypt}^+(pk, vk, m)]$, $\sigma \in [\mathsf{Sign}^+(sk, pk, c)]$, the following two random variables are equally distributed:

$$\left[\begin{array}{l} c' \leftarrow \mathsf{Encrypt}^+(pk, vk, m); \\ \sigma' \leftarrow \mathsf{Sign}^+(sk, pk, c'); \end{array} : (c', \sigma')\right] \approx \left[(c', \sigma') \leftarrow \mathsf{Random}^+(pk, vk, c, \sigma) : (c', \sigma')\right] .$$

Unforgeability. Unforgeability of signatures on randomizable ciphertext is defined via the following experiment: The challenger computes a signature key pair and an encryption key pair (sk, vk), (dk, pk) and runs the adversary on (vk, pk, dk). It is also given access to an oracle $\mathsf{Sign}^+(sk, pk, \cdot)$, which it can query adaptively on ciphertexts $c_1, \ldots, c_q$ of its choice. Finally, the adversary outputs a pair (c^*, σ^*) and wins if $\mathsf{Verify}^+(vk, pk, c^*, \sigma^*) = 1$ and $m = \mathsf{Decrypt}^+(dk, vk, c^*)$ is different from all $m_i := \mathsf{Decrypt}^+(dk, vk, c_i)$.

9.5 Our SRC Construction

At a high level, we need a construction that enforces our restricted message space and is malleable enough to be re-randomised but no more. The first requirement ensures that voters can only submit valid ballots, while the second gives us privacy via randomization while preventing copying or tampering attacks. Specifically, we use GS proofs to ensure validity and prevent copying or producing ballots related to those of another user. We use signatures to ensure integrity, meaning a randomiser cannot change the ballot contents.

9.5.1 Asymmetric Waters signature scheme.

Blazy *et al.* [BFPV11a] define a variant of Waters' signature scheme [Wat05] for asymmetric groups that is perfectly randomizable and which they prove secure under the CDH$^+$ assumption.

$\mathsf{Setup}(1^\lambda, 1^k)$: To sign messages $m = (m_1, \ldots, m_k) \in \{0,1\}^k$, generate $(p, \mathbb{G}_1, \mathbb{G}_2, g_1, g_2, \mathbb{G}_T, e) \leftarrow \mathsf{GrpGen}(1^\lambda)$, choose $z \leftarrow \mathbb{G}_1$, $\boldsymbol{u} = (u_0, \ldots, u_k) \leftarrow \mathbb{G}_1^{k+1}$, define $\mathcal{F}(m) := u_0 \prod_{i=1}^{k} u_i^{m_i}$. Output $pp = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, z, \boldsymbol{u})$.

$\mathsf{SKeyGen}(pp)$: Choose $x \leftarrow \mathbb{Z}_p$, define $X_1 := g_1^x$, $X_2 := g_2^x$, $Y := z^x$; output the public key $vk = (pp, X_1, X_2)$ and the secret key $sk = (pp, Y)$.

$\mathsf{Sign}(sk = (pp, Y), m; s)$: For randomness $s \in \mathbb{Z}_p$, return the signature σ defined as

$$\left(\sigma_1 := Y \cdot \mathcal{F}(m)^s,\ \sigma_2 := g_1^s,\ \sigma_3 := g_2^s\right) .$$

$\mathsf{Verif}(vk = (pp, X_1, X_2), m, \sigma)$: Output 1 if both of the following hold and 0 otherwise:

$$e(\sigma_1, g_2) = e(z, X_2) \cdot e(\mathcal{F}(m), \sigma_3) \qquad e(\sigma_2, g_2) = e(g_1, \sigma_3)$$

$\mathsf{Random}((pp, X_1, X_2), F, \sigma; s')$: For randomness $s' \in \mathbb{Z}_p$, output $\sigma' := (\sigma_1 \cdot F^{s'},\ \sigma_2 \cdot g_1^{s'},\ \sigma_3 \cdot g_2^{s'})$.

Note that for Random it suffices to know the hash $F = \mathcal{F}(m)$ of the signed message. The scheme is perfectly randomizable, as for any $((pp, X_1, X_2), (pp, Y)) \in [\mathsf{SKeyGen}(pp)]$ and m, s, s' we have $\mathsf{Random}((X_1, X_2), \mathcal{F}(m), \mathsf{Sign}(Y, m; s), s') = \mathsf{Sign}(Y, m; s + s')$.

Remark 4. *Blazy et al. [BFPV11a] show that their signature scheme also satisfies a (stronger) EUF-CMA notion, where the adversary's signing queries are of the form* (m, R, T) *and if* $e(T, g_2) = e(R, X_2)$ *then the oracle returns an additional signature element* $\sigma_4 = R^s$.

9.5.2 Our SRC scheme.

We will combine ElGamal encryption, Groth-Sahai proofs and Waters signatures to create an SRC scheme. Our construction extends that of [BFPV11a], so that it immediately yields an RCCA-secure encryption scheme (defined below) and ultimately a strongly receipt-free e-voting scheme.

Our scheme is defined for a polynomial-size message space $\mathcal{M} = \{0, 1\}^k$, that is, we assume k to be logarithmic in the security parameter. Messages m are encrypted as ElGamal ciphertexts of $\mathcal{F}(m)$. Decryption works by decrypting a ciphertext to F and then looking for m with $F = \mathcal{F}(m)$. We define a function H and add a third ciphertext element $c_3 = H(vk)^r$, which will tie the ciphertext to the verification key for which it is produced.

We moreover add C_m, Groth-Sahai commitments to the message bits, and a commitment C_T to X_1^r, which is needed for the security reduction (it corresponds to T from Remark 4). Finally, we add GS proofs which show consistency of these commitments, and consistency of the additional ciphertext element c_3. In more detail, in order to show a component C_{m_i} of C_m contains a bit, we require commitments in both groups $(C_{1,m,i}, C_{2,m,i})$ and proofs π'_i. A commitment C_r to the randomness r is used to prove consistency of the values c_1 with C_r, c_2 with C_r and $\{C_{m,i}\}_{i=1}^{k}$, c_3 with C_r, as well as C_T with X_1 and C_r.

Now, given ciphertext elements $c_1 = g_1^r$ and $c_2 = \mathcal{F}(m) \cdot P^r$, the crucial observation is that, due to the interoperability of ElGamal encryption and Waters signatures, a signer can produce an encryption of a signature on the *plaintext*, without knowing the latter: setting $\sigma_1" = c_1^s = g_1^{rs}$ and $\sigma_2 := Y \cdot c_2^s = Y \cdot \mathcal{F}(m)^s \cdot P^{rs}$ yields an encryption under P of the first Waters signature element $Y \cdot \mathcal{F}(m)^s$. This is completed to a full signature by (g_1^s, g_2^s). Finally, in order to enable full randomization of ciphertext/signature pairs, we also include P^s in the signature.

Let Setup (for Waters signatures), $\mathsf{Setup}_{\mathsf{GS}}$ (for Groth-Sahai proofs) and KeyGen (for ElGamal encryption) be defined as above, and $H\colon \{0,1\}^\star \to \mathbb{G}_1$ be defined as

$$H(x) := h_1 \cdot h_2^{H'(vk)} \tag{9.1}$$

for $h = (h_1, h_2) \in \mathbb{G}_1^2$ and a collision-resistant hash function $H'\colon \{0,1\}^\star \to \mathbb{Z}_p$. Our scheme is given in Figure 9.2. It is based on the scheme from [BFPV11a], to which we add the crucial ciphertext elements c_3 and π_V. Unforgeability still holds, as the reduction can program the random oracle so it knows the logarithm of $H(vk)$, and thus simulate c_3.

We omit the specific structure of the proofs in $\boldsymbol{\pi}$ as they will not be relevant to the rest of this work. Informally, π_m ensures the message is valid, and π_T and π_V tie the ciphertext to vk. Under a binding CRS, the bitwise commitments enable us to extract the message even in the absence of the decryption key.

Theorem 17. *The SRC scheme* $(\mathsf{Setup}, \mathsf{EKeyGen}, \mathsf{SKeyGen}, \mathsf{Encrypt}^+, \mathsf{Decrypt}^+, \mathsf{Sign}^+, \mathsf{Verify}^+, \mathsf{Random}^+)$ *defined in Figure 9.2 is unforgeable under the CDH$^+$ assumption.*

Remark 5. *We will prove a stronger statement, namely that our SRC scheme is unforgeable even when the adversary only needs to output a "partial" forgery* $(c_1, c_2, \{C_{1,m,i}, C_{2,m,i}\}, C_r, \pi_r, \pi_m), (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$*, i.e., it need not contain* c_3, C_T, π_T, π_V *and* σ_5.

Moreover, note that one can also decrypt ciphertexts using the extraction trapdoor ξ for GS proofs to recover m from C_m, sidestepping the inefficient hash inversion. We let $\mathsf{EKeyGen}^{(x)}$ *denote key generation that returns ξ instead of* dk.

```
$\mathsf{EKeyGen}(1^\lambda, 1^k)$:
    $pp := (\mathcal{G}, z, \boldsymbol{u}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k)$
    $crs \leftarrow \mathsf{Setup}_{\mathsf{GS}}(\mathcal{G})$; $h \leftarrow \mathbb{G}_1^2$
    $(P, dk) \leftarrow \mathsf{KeyGen}(\mathcal{G}, 1)$
    return $pk := (pp, crs, h, P)$, $dk$.

$\mathsf{Encrypt}^+((pp, crs, h, P), vk = (pp, X_1, X_2), m; r)$:
    Compute, with $H$ defined by $h$ as in (9.1):
        $c_1 := g_1^r$        $c_2 := \mathcal{F}(m) \cdot P^r$
        $c_3 := H(vk)^r$
    Make commitments $\boldsymbol{C}$: For $i = 1, \dots, k$:
        $C_{1,m,i} := \mathcal{C}_1'(m_i)$    $C_{2,m,i} := \mathcal{C}_2'(m_i)$
        $C_T := \mathcal{C}_1(X_1^r)$    $C_r := \mathcal{C}_2'(r)$
    Compute GS proofs $\boldsymbol{\pi}$ for the following
    (with $\overline{r}$ being the value committed in $C_r$;
    $\overline{m}_i$ in $C_{2,m,i}$; and $\overline{w}$ in $C_T$):
        • $\pi_r$ proves $g_1^{\overline{r}} = c_1$.
        • $\pi_m$ consists of:
              $\pi_i'$ proving $\overline{m}_i$ is a bit for all $i$;
              $\pi_m'$ proving $c_2 = u_0 \prod_{i=1}^k u_i^{\overline{m}_i} \cdot P^{\overline{r}}$.
        • $\pi_T$ proves $X_1^{\overline{r}} = \overline{w}$.
        • $\pi_V$ proves $H(vk)^{\overline{r}} = c_3$.
    Return $c := (c_1, c_2, c_3, \boldsymbol{C}, \boldsymbol{\pi})$.

$\mathsf{Sign}^+(sk = (pp, Y), (pp, crs, h, P), c; s)$:
    If $\boldsymbol{\pi}$ is not valid for $\boldsymbol{C}, vk, P$, return $\perp$.
    Else return
        $\sigma_1 := c_1^s$    $\sigma_2 := Y \cdot c_2^s$        (9.2)
        $\sigma_3 := g_1^s$    $\sigma_4 := g_2^s$    $\sigma_5 := P^s$

$\mathsf{Random}^+(vk, (pp, crs, h, P), c, \sigma; (r', s'))$:
    Let $c = (c_1, c_2, c_3, \boldsymbol{C}, \boldsymbol{\pi})$; set:
        ${c_1}' := c_1 \cdot g_1^{r'}$        ${c_2}' := c_2 \cdot P^{r'}$
        ${c_3}' := c_3 \cdot H(vk)^{r'}$
        ${\sigma_1}' := \sigma_1 \cdot c_1^{s'} \cdot \sigma_3^{r'} \cdot g_1^{r' \cdot s'}$
        ${\sigma_2}' := \sigma_2 \cdot c_2^{s'} \cdot \sigma_5^{r'} \cdot P^{r' \cdot s'}$
        ${\sigma_3}' := \sigma_3 \cdot g_1^{s'}$        ${\sigma_4}' := \sigma_4 \cdot g_2^{s'}$
        ${\sigma_5}' := \sigma_5 \cdot P^{s'}$
    Set $C_r' = C_r \cdot \mathcal{C}_2(r')$, adapt $\pi_r, \pi_T, \pi_V$ accordingly.
    Randomise all commitments and proofs to $\boldsymbol{C}'$ and $\boldsymbol{\pi}'$.
    Return $(c_1', c_2', c_3', \boldsymbol{C}', \boldsymbol{\pi}')$ and $\sigma'$.

$\mathsf{Verify}^+((pp, X_1, X_2), (pp, crs, h, P), (c_1, c_2, c_3, \boldsymbol{C}, \boldsymbol{\pi}), \sigma)$:
    Return 1 iff $\boldsymbol{\pi}$ verifies and the following hold:
        $e(\sigma_1, g_2) = e(c_1, \sigma_4)$                        (9.3a)
        $e(\sigma_2, g_2) = e(z, X_2)\, e(c_2, \sigma_4)$            (9.3b)
        $e(\sigma_3, g_2) = e(g_1, \sigma_4)$    $e(\sigma_5, g_2) = e(P, \sigma_4)$
                                                                     (9.3c)

$\mathsf{Decrypt}^+(dk, (pp, crs, h, P), vk, c)$:
    Let $c = (c_1, c_2, c_3, \boldsymbol{C}, \boldsymbol{\pi})$.
    If $\boldsymbol{\pi}$ is not valid, return $\perp$
    Else let $F := \mathsf{Decrypt}(dk, c = (c_1, c_2))$;
    browse $\mathcal{M}$ and return the first $m$ with $\mathcal{F}(m) = F$.
```

Figure 9.2: Our SRC scheme

Proof. The proof is by reduction from unforgeability of Waters signatures. The reduction obtains a verification key vk including parameters pp. It simulates $\mathsf{EKeyGen}$ by running $(crs, \xi) \leftarrow \mathsf{Setup}_{\mathsf{GS}}^{(x)}(\mathcal{G})$, $(P, d) \leftarrow \mathsf{KeyGen}(\mathcal{G}, 1)$ and choosing $h \leftarrow \mathbb{G}_1^2$, and runs the adversary on $vk, pk := (pp, crs, h, P)$ and $dk := d$. If the adversary queries a signature on a valid tuple $c = (c_1, c_2, c_3, \boldsymbol{C}, \boldsymbol{\pi})$, the reduction uses ξ to extract m and $T = X_1^r$ from $\boldsymbol{C}$ (note that by soundness of $\boldsymbol{\pi}$, we have $m = \mathsf{Decrypt}^+(dk, vk, c)$). The reduction makes a special query, as defined in Remark 4, (m, c_1, T) to its signing oracle (note that c_1, T satisfy $e(T, g_2) = e(c_1, X_2)$); it obtains a signature $(\tau_1 = Y\mathcal{F}(m)^s, \tau_2 = g_1^s, \tau_3 = g_2^s, \tau_4 = c_1^s)$; it defines (letting r be the unknown randomness in (c_1, c_2, c_3)) $\sigma_1 := \tau_4 = g_1^{rs}$, $\sigma_2 := \tau_1 \cdot \tau_4^d = Y\mathcal{F}(m)^s P^{rs}$, $\sigma_3 := \tau_2 = g_1^s$, $\sigma_4 := \tau_3 = g_2^s$, $\sigma_5 := \tau_2^d = P^s$, which is distributed as an SCR signature on c.

Let $\{m_1, \dots, m_q\}$ be the extracted (equivalently: decrypted) messages of the signing queries. Assume the adversary outputs a (partial) valid forgery, namely one which only contains $(c_1, c_2, \sigma_1, \sigma_2, \sigma_3, \sigma_4)$, commitments C_m, C_r and proofs π_r, π_m. The reduction extracts m from C_m. Then soundness of π_r and π_m ensures that for some r we have $c_1 = g_1^r$ and $c_2 = \mathcal{F}(m)P^r$ (and thus $m = \mathsf{Decrypt}^+(dk, vk, c)$).

Moreover, let s be such that $\sigma_4 = g_2^s$. Since the forgery is valid, from Verify^+ we have: $\sigma_1 = g_1^{rs}$ (from (9.3a)), $\sigma_2 = Y\mathcal{F}(m)^s P^{rs}$ (from (9.3b)) and $\sigma_3 = g_1^s$ (from (9.3c)). The reduction sets $\sigma_1^* := \sigma_2 \cdot \sigma_1^{-d} = Y\mathcal{F}(m)^s$, $\sigma_2^* := \sigma_3 = g_1^s$ and $\sigma_3^* := \sigma_4 = g_2^s$ and returns (m, σ^*). This is a valid Waters forgery, as σ^* is valid for m and $m \notin \{m_1, \dots, m_q\}$ (otherwise the adversary would not have won the SRC unforgeability game). $\square$

9.5.3 RCCA-Secure Encryption from SRC

As a next step towards our voting protocol, we show that our SRC scheme, contrary to the one from [BFPV11a], yields an RCCA-secure [CKN03] encryption scheme, as defined next.

CCA-security (Sect. 3.3.6) is the standard notion for public-key encryption and implies that ciphertexts are non-malleable. It states that for an efficient adversary which after choosing m_0, m_1 receives c^* it should be impossible to decide whether c^* encrypts m_0 or m_1, even when given an oracle that decrypts any ciphertext $c \neq c^*$. For randomizable schemes this notion is unachievable, as the adversary could submit a randomization of the challenge ciphertext to the decryption oracle. The strongest achievable notion for randomizable schemes is RCCA, where whenever the oracle receives an encryption of m_0 or m_1, it returns a special symbol $\top$.

Based on our SRC scheme (Sect. 9.5.2) we define the following candidate RCCA encryption scheme for a polynomial-size message space $\{0,1\}^k$, and show it to be secure. The construction is a shim designed to adapt SRC to RCCA which is necessary as SRC and RCCA have different functionalities.

$\overline{\mathsf{KeyGen}}$ is defined as $\mathsf{EKeyGen}$.

$\overline{\mathsf{Encrypt}}(pk, m)$: Run $(vk, sk) \leftarrow \mathsf{SKeyGen}(pp)$; $c \leftarrow \mathsf{Encrypt}^+(pk, vk, m)$; $\sigma \leftarrow \mathsf{Sign}^+(sk, pk, c)$; return $\bar{c} = (c, \sigma, vk)$.

$\overline{\mathsf{Decrypt}}(dk, (c, \sigma, vk))$: If $\mathsf{Verify}^+(vk, pk, c, \sigma) = 1$, return $m := \mathsf{Decrypt}^+(dk, vk, c)$; else return $\perp$.

$\overline{\mathsf{Random}}$ is defined as Random^+.

The adversary is free to create fresh vk, sk and run $\overline{\mathsf{Encrypt}}$ itself or use it as an oracle (representing an attack against users with unknown secret keys).

Theorem 18. *The above encryption scheme for polynomial-size message spaces is RCCA-secure under the SXDH and the CDH$^+$ assumption.*

Proof. Intuitively, ciphertexts hide the message, since under SXDH we could replace the commitments and proofs in the challenge ciphertext by simulated ones and under DDH, we could replace $c_2 = \mathcal{F}(m)P^r$ by a random element, so the ciphertext would contain no more information about the message. The difficulty is that we need to simulate the decryption oracle. For this we program the hash function H: let vk^* be the key contained in the challenge ciphertext; we choose $a, b \leftarrow \mathbb{Z}_p$ and set $h_1 = P^{-a \cdot H'(vk^*)} \cdot g_1^b$ and $h_2 = P^a$, which is distributed correctly and set $H(vk) = P^{a(H'(vk)-H'(vk^*))} \cdot g_1^b$. For a well-formed ciphertext containing $vk_i \neq vk^*$, we then have $c_2 \cdot (c_3 \cdot c_1^{-b})^{-1/(a(H'(vk)-H'(vk^*)))} = c_2 \cdot P^{-r} = c_2 \cdot c_1^{-d} = \mathsf{Decrypt}(d, (c_1, c_2))$, meaning we can use c_3 to decrypt without knowing d; for the challenge ciphertext under vk^* we have $c_3 = g_1^{br}$, so we can embed a DDH challenge.

The reduction can thus answer decryption queries containing some $vk \neq vk^*$, but not if it contains vk^*. However, if an adversary submits a valid ciphertext with vk^* which does not encrypt the challenge message, then it would break SRC unforgeability, so security of our SRC scheme implies that the adversary cannot make this type of query. □

9.6 BeleniosRF

In this section we define Belenios Receipt-Free (BeleniosRF), a strongly receipt-free voting protocol that builds on [BFPV11a, CGGI14].

9.6.1 Overview

The election public/secret key pair $(\mathbf{pk}, \mathbf{sk})$ is an encryption/extraction key pair generated via $\mathsf{EKeyGen}^{(x)}$ (i.e the extractable variant of the key genarator – cf. Remark 5), and user key pairs (upk, usk) are signature keys generated by $\mathsf{SKeyGen}$. A user casts a vote by encrypting it via $\mathsf{Encrypt}^+$ under $\mathbf{pk}$ w.r.t. his upk, and uses usk to then sign the ciphertext via Sign^+ (together, this corresponds to a ciphertext of our RCCA encryption scheme from Section 9.5.3).

When the ballot box receives a valid ballot, it randomises it via Random^+ and immediately publishes the resulting ciphertext/signature pair on the public bulletin board PBB.

Users can verify that their vote is present, since they can verify the adaptation of their signature on their now-randomised ciphertexts.

Tallying follows standard techniques of e-voting: our construction allows for homomorphic tallying as well as shuffling. In the first case, we take advantage of the special structure of GS commitments, which allow us to calculate a partial tally for each option by adding the corresponding commitment across voters, and then decrypting the resulting commitment (with proof of correctness).

Using shuffling, the encrypted votes are re-randomised and shuffled (and a proof of correct execution of this is generated) via an algorithm $\mathsf{Shuffle}$. Then the ballots are decrypted (again accompanied with a proof that this was done correctly) and the result is published. These proofs make the tallying process publicly verifiable.

We now describe the homomorphic tallying version, where $\mathbb{V} = \{0,1\}^k$, and the result function is simple vector addition.

The scheme $\mathcal{V}^{\mathsf{BeleniosRF}}$ is based on the SCR scheme from Section 9.5 and consists of the following algorithms:

$\mathsf{Setup}(1^\lambda, 1^k)$: Compute $(\mathbf{pk}, \mathbf{sk}) \leftarrow \mathsf{EKeyGen}^{(x)}(1^\lambda, 1^k)$, produce a Fiat-Shamir random oracle proof Π_σ that *crs*, contained in $\mathbf{pk}$, is binding. Return $(\mathbf{pk}^* = (\mathbf{pk}, \Pi_\sigma), \mathbf{sk})$.

$\mathsf{Register}(id)$: On (implicit) input $\mathbf{pk} = (pp, crs, h, P)$, the user runs $(upk_{id}, usk_{id}) \leftarrow \mathsf{SKeyGen}(pp)$ and produces π_{id}, a Fiat-Shamir random oracle proof proof that user *id* knows usk_{id} corresponding to upk_{id}. The registrar checks $\pi_i d$ and if it is valid and $upk_{id} \notin L$ it adds upk_{id} to L.

$\mathsf{Vote}(id, upk, usk, v)$ is used by a voter to create a ballot b for vote $v \in \mathbb{V}$. It computes $c \leftarrow \mathsf{Encrypt}^+(\mathbf{pk}, upk, v)$ and $\sigma \leftarrow \mathsf{Sign}^+(usk, \mathbf{pk}, c)$; and returns $b := (id, upk, c, \sigma)$.

$\mathsf{Valid}(\mathsf{BB}, b)$ first checks that the ballot b is *valid*, i.e., that it is well-formed and the signature is correct. Formally, it parses b as (id, upk, c, σ) and checks if

– *id* corresponds to an eligible voter from *ID* and *upk* corresponds to the registration of user *id*;

– $\mathsf{Verify}^+(upk, \mathbf{pk}, c, \sigma) = 1$.

If any step fails, it returns $\perp$; otherwise, it returns $\top$.

$\mathsf{Append}(\mathsf{BB}, b = (id, upk, c, \sigma))$ randomises (c, σ) as $(c', \sigma') \leftarrow \mathsf{Random}^+(upk, \mathbf{pk}, c, \sigma)$ and appends to BB a randomised version $b' = (id, upk, c', \sigma')$ of b.

$\mathsf{Publish}(\mathsf{BB})$ takes every entry $b{=}(id, upk, c, \sigma)$ in BB and removes elements $id, c_3, C_T, \pi_T, \pi_V$ and σ_5, constructing $\hat{b} := \big(upk, (c_1, c_2, C_m, C_r, \pi_r, \pi_m), (\sigma_1, \sigma_2, \sigma_3, \sigma_4)\big)$. It then adds $\hat{b}$ to PBB,[2] and returns PBB.

$\mathsf{VerifyVote}(\mathsf{PBB}, id, upk, usk, b)$ browses PBB for an entry $\hat{b}$ containing upk. If none exists, it returns $\bot$. For entry $\hat{b} := \big(upk{=}(pp, X_1, X_2), (c_1, c_2, C_m, C_r, \pi_r, \pi_m), (\sigma_1, \sigma_2, \sigma_3, \sigma_4)\big)$ if π_r and π_m are valid and

$$e(\sigma_1, g_2) = e(c_1, \sigma_4) \qquad e(\sigma_2, g_2) = e(z, X_2) \cdot e(c_2, \sigma_4) \qquad e(\sigma_3, g_2) = e(g_1, \sigma_4)$$

then return $\top$, else return $\bot$.

$\mathsf{Tally}(\mathsf{BB}, \mathbf{sk})$ consists of the following steps. Let N be the number of ballots.

- Parse each ballot $b \in \mathsf{BB}$ as $b = (id^{(b)}, upk^{(b)}, c^{(b)}, \sigma^{(b)})$.
- If there is any ballot b that does not pass $\mathsf{Valid}(\mathsf{BB}, b)$, output $(r = \bot, \mathsf{PBB}, \Pi_d = \emptyset)$.
- Let $\{C^{(b)}_{1,m,i}\}_{i=1}^{k}$ be the commitments in $C^{(b)}_m$ contained in $c^{(b)}$. Compute $T_i := \sum_{b \in \mathsf{BB}} C^{(b)}_{1,m,i}$. The tally t_i for candidate i is produced by decrypting T_i with the GS extraction key $\xi = \mathbf{sk}$.
- Produce the result $r = (t_1, \dots, t_k)$ and Π_d, a Fiat-Shamir proof of correct extraction.
- Output (r, PBB, Π_d).

$\mathsf{Verify}(\mathsf{PBB}, r, \Pi_\sigma, \Pi_d)$ simply verifies Π_σ w.r.t. crs and Π_d w.r.t. PBB and the result r.

9.6.2 Receipt-Freeness

We now show that BeleniosRF satisfies strong receipt-freeness, as defined in Definition 30. Note that this in particular implies vote privacy of BeleniosRF.

Theorem 19. *$\mathcal{V}^{\mathsf{BeleniosRF}}$ is strongly receipt-free under the SXDH assumption in the random-oracle model.*

Proof. The proof uses the ideas of that of Theorem 18. The main one is again to use hash-function programmability and to decrypt a ciphertext (c_1, c_2, c_3) using components c_2 and c_3 instead of the GS commitments. This will allow us to switch to a hiding CRS, for which the commitments would not be extractable. By randomizability of our SCR scheme and of

[2] As noted in Remark 5, these are precisely the elements that guarantee unforgeability, which assures a voter that the plaintext of his encrypted vote was not altered.

Groth-Sahai proofs, instead of re-randomizing the ballots in PBB, we can simply recompute them. Finally, having switched to a hiding CRS and a simulated ROM proof thereof, we are able to replace the adversary's view with uniformly distributed values, irrespective of β.

We proceed by a sequence of hybrid games, which we show are indistinguishable:

Hybrid ($\boldsymbol{\beta}$,0) is the sRF game $\mathsf{Exp}^{\mathsf{srf},\beta}_{\mathcal{A},\mathcal{V}}$ (Definition 30 and Figure 9.1).

Hybrid ($\boldsymbol{\beta}$,1) is the same game as Hybrid (β,0) for $\beta = 1$; for $\beta = 0$ the difference is that the Fiat-Shamir proofs for the CRS and the tally are simulated.

Hybrid (β,0) $\rightarrow$ Hybrid (β,1): Since ROM proofs can be perfectly simulated by using random-oracle programmability, the two hybrid games are distributed equivalently.

Hybrid ($\boldsymbol{\beta}$,2) is defined as Hybrid (β,1), except for how h is chosen. For $a, b \leftarrow \mathbb{Z}_p$ we define $h_1 := g_1^b$ and $h_2 := P^a$ (as in Theorem 18 but setting $H'(vk^*) := 0$).

Hybrid (β,1) $\rightarrow$ Hybrid (β,2): It is immediate that both games are distributed equivalently.

Hybrid ($\boldsymbol{\beta}$,3) is defined as Hybrid (β,2), but the result is computed differently: each ballot $b_i = (id_i, upk_i, c_i, \sigma_i)$ is decrypted as $F_i := c_{i,2} \cdot (c_{i,3} \cdot c_{i,1}^{-b})^{-1/(a \cdot \mathsf{upk}_i)}$ and vote v_i is defined as the smallest $v_i \in \{0,1\}^k$ satisfying $\mathcal{F}(v_i) = F_i$. The result is $r = (t_1, \ldots, t_k)$ with $t_j = \sum_i v_{i,j}$.

Hybrid (β,2) $\rightarrow$ Hybrid (β,3): Perfect soundness of the GS proofs contained in c_i guarantees that this alternative way of decryption leads to the same result as extracting the bits of v_i from the commitments (we ignore collisions in $\mathcal{F}$ which only occur with negligible probability).

Hybrid ($\boldsymbol{\beta}$,4) is defined as Hybrid (β,3), except that PBB is computed differently: for ballot b_i, after extracting v_i, instead of re-randomizing b_i, we freshly compute $\hat{b}_i$ for user i with $usk_i = (pp, Y_i)$ as follows: we pick $r_i, s_i \leftarrow \mathbb{Z}_p$ to set

$$c_{i,1} := g_1^{r_i} \quad c_{i,2} := \mathcal{F}(v_i) \cdot P^{r_i} \quad \sigma_{i,1} := c_1^{s_i} \quad \sigma_{i,2} := Y_i \cdot c_2^{s_i} \quad \sigma_{i,3} := g_1^{s_i} \quad \sigma_{i,4} := g_2^{s_i}$$

and using witnesses r_i and v_i, we compute $C_{i,m}, C_{i,r}$ and $\pi_{i,r}, \pi_{i,m}$. We set $\hat{b}_i = \big(upk_i, (c_{i,1}, c_{i,2}, C_{i,m}, C_{i,r}, \pi_{i,r}, \pi_{i,m}), (\sigma_{i,1}, \sigma_{i,2}, \sigma_{i,3}, \sigma_{i,4})\big)$.

Hybrid (β,3) $\rightarrow$ Hybrid (β,4): By re-randomizability of our SCR scheme and GS proofs, re-randomised ciphertexts, signatures and proofs are distributed exactly as freshly computed ones. The two hybrids are thus equally distributed.

Hybrid ($\boldsymbol{\beta}$,5) is defined as Hybrid (β,4), except that the CRS contained in **pk** is set up in hiding mode, i.e., computed via $\mathsf{Setup}_{\mathrm{GS}}^{(h)}$.

Hybrid (β,4) $\rightarrow$ Hybrid (β,5): By the properties of GS proofs, the two hybrids are indistinguishable under the SXDH assumption.

Hybrid ($\boldsymbol{\beta}$,6) is defined as Hybrid (β,5), except that the commitments and proofs published in PBB are simulated.

Hybrid (β,5) $\rightarrow$ Hybrid (β,6): By the properties of GS proofs, under a hiding CRS regularly computed proofs and simulated proofs are distributed equivalently; the two hybrids are thus equally distributed.

Hybrid ($\boldsymbol{\beta}$,7) is defined as Hybrid (β,6), except that for every i, when computing PBB entry $\hat{b}_i$, $c_{i,2}$ is computed as $c_{i,2} := \mathcal{F}(v_i) \cdot g_1^{w_i}$ for $w_i \leftarrow \mathbb{Z}_p$.

Hybrid (β,6) $\rightarrow$ Hybrid (β,7): The two hybrids are indistinguishable under the DDH assumption in $\mathbb{G}_1$, which is proved as follows: we first note that in Hybrid (β,6), d (the decryption key with $P = g_1^d$) is not used anywhere, and r_i is only used to compute $c_{i,1}$ and $c_{i,2}$ (since the GS commitments and proofs are simulated).

We give a reduction from DDH to distinguishing Hybrids 6 and 7. Let $(P = g_1^d, R = g_1^r, W)$ be a DDH instance, where either W is random or $W = g_1^{d \cdot r}$. By random self-reducibility of DDH [BBM00] we can create arbitrarily many instances (P, R_i, W_i), where $R_i = g_1^{r_i}$ for some uniformly random r_i, and W_i is independently random if W was, or $W_i = g_1^{d \cdot r_i}$ if $W = g_1^{d \cdot r}$.

The simulator now sets $\mathbf{pk} := (pp, crs, P)$, with P from the instance, and $c_{i,1} := R_i$ and $c_{i,2} := \mathcal{F}(v_i) \cdot W_i$. If $W_i = P^{r_i}$ then this is distributed as in Hybrid (β,6), whereas if W_i is random, this is distributed as in Hybrid (β,7).

Observe that Hybrid (0,7) and Hybrid (1,7) are equally distributed, since in both games every ciphertext $(c_{i,1}, c_{i,2})$ is a uniformly random pair. We have thus constructed a sequence of hybrid games Hybrid (0,0), ..., Hybrid (0,7), Hybrid (1,7), ..., Hybrid (1,0) which are indistinguishable under SXDH and of which the first one corresponds to the sRF game with $\beta = 0$ and the last is the sRF game with $\beta = 1$. This concludes the proof of strong receipt-freeness of BeleniosRF. $\square$

Remark 6. *We note that our scheme can be easily modified and proven secure in the standard model if we assume a trusted CRS: drop Π_σ in* Setup *and use GS proofs for Π_d.*

9.6.3 Verifiability

We consider strong verifiability from [CGGI14], which intuitively ensures that the result of the election reflects the votes of:

- All voters who properly checked that their ballot appears in the bulletin board at the end of the election. In BeleniosRF, a voter should check that one ballot in PBB is signed with her credential.
- A subset of the voters who did not perform that final check. A voters may stop after casting her vote, thus there is no guarantee that her ballot made it into the ballot box. However, if the ballot is present, it should not be possible to *modify* the corresponding vote.
- At most all corrupted voters. In particular, an adversary should not be able to add more votes than the number of voters he controls.

We refer the reader to [CGGI14] for the formal definition and point out that strong verifiability assumes that voting devices are honest. We first note that BeleniosRF cannot be strongly verifiable if revoting is allowed. Indeed, if a voter first casts a ballot b_1 for a candidate v_1, but later changes her mind and votes for v_2, casting a new ballot b_2, a malicious voting server may force the voter to keep the initial vote v_1 by re-randomizing b_1 instead of b_2, and the voter would not be able to detect it. Therefore, in what follows, we assume that a no-revote policy is applied. We believe that no-revoting is not a real restriction since, as discussed in the introduction, this actually corresponds to the most common setting used in practice. By slightly generalizing the strong-verifiability transformation in [CGGI14, Section 4], we are able to show:

Theorem 20. *BeleniosRF is strongly verifiable, if the underlying signature on randomizable ciphertexts scheme is unforgeable and either: (a) the ballot box is honest or (b) the registration protocol is zero-knowledge.*

Let us summarise the main ideas behind our argument. The transformation to strong verifiability in [CGGI14] consists in the voter signing with her private signing key usk a ballot b obtained via an existing voting protocol that is *weakly verifiable* (roughly speaking, weak verifiability assumes that the voting server is honest, e.g., it does not modify nor erase ballots). Next, the voter sends the triple (upk, b, σ) to the voting server. The latter, after validating the ballot b and verifying its signature σ, adds the triple (upk, b, σ) to the ballot box. At the end of the election, the voter checks that her ballot (upk, b, σ) appears in PBB by a simple search. (i.e. she checks that $(upk, b, \sigma) \in$ PBB).

We generalise this transformation by allowing the voting server to add a *transformed* triple (upk, b', σ') to the ballot box on input the voter's ballot (upk, b, σ), such that potentially $b \neq b'$ and $\sigma \neq \sigma'$ (in the original construction, one simply sets $b' = b$ and $\sigma' = \sigma$). In our generalised transformation, the voter on input her cast ballot (upk, b, σ) checks whether there exists an entry (upk, b', σ') in PBB such that (b', σ') verifies under her key upk. Due to unforgeability of randomizable signatures on ciphertexts (cf. Section 9.5) and because of the no-revoting policy, this check guarantees that the new ballot b' displayed in the bulletin board contains the same vote as the original ballot b cast by the voter.

If we opt to use an issuing (as opposed to an endorsing) registar, strong verifiability assumes that either the ballot box (i.e. the re-randomization server) or the registrar is honest. As pointed out in Section 9.2, the security of the generalised transformation described in the previous paragraph is jeopardised if this trust assumption is violated, as the existence of an entry (upk, b', σ') in PBB would no longer guarantee that b' contains the choice cast by the voter. In fact, an attacker controlling both an *issuing* registar and the voting server can insert entries (upk, b', σ') in PBB that pass all tests but modified the voter's choice. This is due to the fact that the registrar knows each voter's private signing key. An obvious countermeasure is to let each voter generate their own signing key pair and simply ask the registrar to include the corresponding verification key in the list of eligible keys for the election.

Alternatively, one can thresholdise the role of the registrar (who simply sends a private signing key to each voter) so it becomes less likely for the attacker to obtain a voter's private key.

9.7 Efficiency of BeleniosRF

The ballot encryption scheme we introduced is somewhat involved, especially since we use bit-by-bit Groth-Sahai proofs. For this reason, we benchmarked ballot creation on a number of potential client devices. We built a JavaScript implementation [bel16] of the voting process (encrypt, sign, prove) using the CertiVox IoT Crypto Library [Cer15]. We used a BN curve on a 254-bit prime field. We considered the values $k = 1, 5, 10$ and 25. For homomorphic tallying, as used in Section 9.6, k represents the number of candidates in an election. If we switch to shuffle-based tallying, k is the length of the message, which means we can support up to 2^k candidates.

As seen in Table 9.1, recent devices can complete the required cryptographic operations in reasonable time for small values of k. We see that while the linear cost associated with the message size is the dominant factor, the constant factor is not negligible for low-end devices. While slower than the current Helios or Belenios implementation (which do not use elliptic curves), performance is acceptable, especially for modern devices. Moreover, we note that our implementation is single-threaded with only rudimentary optimizations (e.g taking advantage of the fact that the message bits must be 0 or 1). By constructing proofs incrementally as the ballot is filled, we could amortise the linear part of the cost. Alternatively, we may increase performance by coding a native client, e.g. a smartphone app.

We expect that server performance for BeleniosRF will be less of a bottleneck. Compared to Helios, the main additional cost is verifying our Groth-Sahai proofs, which is dominated by ca. $64(k+1)$ pairings, reducible to $4(k+35)$ using techniques from [BFI+10] and [GHR15]. Given the timings provided by Beuchat *et al.* [BGDM+10], we expect a throughput of roughly 5 ballots/second/core for $k = 10$. Additionally, checks can be amortised throughout the voting period, as ballots come in.

9.8 The Blazy *et al.* Voting Protocol is not Ballot-Private

Blazy *et al.* [BFPV11a], who introduced the notion of signatures on randomizable ciphertexts, proposed to use this primitive for a receipt-free e-voting protocol. Their ballot-creation and -casting protocol workflow is as follows:

- The voter sends ballot

$$b = \left(vk,\ c = \{v\}_{pk}^{r},\ \sigma_c^{vk,s}, \pi_{pk}^{t,v\in\{0,1\}}\right),$$

Device	$k=1$	$k=5$	$k=10$	$k=25$
2013 Laptop –i7-4650U	1.00s	2.43s	4.02s	9.24s
2010 Desktop –i3-530	1.49s	3.46s	5.92s	13.62s
2014 Tablet –Exynos 5420	6.97s	12.91s	21.92s	47.26s
2016 Phone –SD 810	2.75s	6.26s	10.39s	22.19s
2014 Phone –SD 801	5.55s	13.12s	22.70s	48.06s
2012 Phone –A6	9.04s	18.65s	29.96s	63.77s

Table 9.1: Time to encrypt, sign and perform GS proofs for ballots with a k-bit payload. This allows for up to k candidates with homomorphic tallying, or 2^k using shuffles.

where $r, s, t \in \mathbb{Z}_p$ denote the randomness used for encrypting the vote v, signing the resulting ciphertext c and creating the NIZK proof π, respectively.

- The server re-randomises the ballot b to b' as follows:
$\big(vk,\, c = \{v\}^{r'}_{pk},\, \sigma_c^{vk,s'},\, \pi_{pk}^{t', v\in\{0,1\}}\big)$, where $r', s', t' \leftarrow \mathbb{Z}_p$.

Similarly to BeleniosRF, the server can only re-randomise legitimate signatures, meaning that any new ballot b' that contains a valid signature w.r.t. vk must originate from a ballot b that has been previously created by the voter, and thus b and b' contain the same vote.

An attack on ballot privacy. However, the above ballot casting workflow is not ballot private, let alone receipt-free. The following is a ballot replay attack, which is known to break ballot privacy [CS11]:

- Honest voter sends $b = \big(vk,\, c = \{v\}^{r}_{pk},\, \sigma_c^{vk,s}, \pi_{pk}^{t, v\in\{0,1\}}\big)$.
- Server re-randomises the ballot b as

  $b' = \big(vk,\, c = \{v\}^{r'}_{pk},\, \sigma_c^{vk,s'},\, \pi_{pk}^{t', v\in\{0,1\}}\big)$

  and displays it on the public bulletin board.
- Dishonest voter with credentials $(\bar{vk}, \bar{sk})$ and knowledge of target ballot b'
  - Copies $c = \{v\}^{r'}_{pk},\, \pi_{pk}^{t', v\in\{0,1\}}$ and re-randomises it to $\bar{c} = \{v\}^{\bar{r}}_{pk},\, \pi_{pk}^{\bar{t}, v\in\{0,1\}}$;
  - Signs $\bar{c}$ with $\bar{sk}$ yielding $\sigma_{\bar{c}}^{\bar{vk},\bar{s}}$;
  - Sends ballot $\bar{b} = \big(vk,\, \bar{c} = \{v\}^{\bar{r}}_{pk},\, \sigma_{\bar{c}}^{\bar{vk},\bar{s}},\, \bar{\pi}_{pk}^{\bar{t}, v\in\{0,1\}}\big)$.

These instructions allow any voter with knowledge of a ballot b to produce an independent-looking ballot $\bar{b}$, that will be accepted by the voting server and effectively contains a copy of the vote in b. Thus, the voting protocol [BFPV11a] is not ballot-private. (Note that due to re-randomizing being allowed, the ballot box cannot discard copied votes, as they look like legitimate ones.)

9.9 Conclusions

We introduced the notion of *strong receipt-freeness*, where a malicious voter (i.e. vote-selling or coerced) cannot produce a receipt proving how she voted, whether the voter decided to act maliciously before, during or after casting the ballot. Even if the voter reveals to the coercer/vote-buyer the randomness used to seal the vote, the adversary cannot be convinced that the revealed coins correspond to the actual randomness used by the voter.

Our adversarial model is close to the spirit of the seminal work on receipt-freeness by Benaloh and Tuinstra [BT94]. Moreover, our definition builds on the recent work [BCG$^+$15], inheriting a simple, concise, and game-based definition. Such definitions are well-known for easing the job of conceiving and writing security proofs; a point we confirm by giving a new e-voting protocol that satisfies our definition in bilinear groups under the SXDH assumption in pairing groups, in the random oracle model. The protocol is built using ideas from a previous work [BFPV11a] that claimed to have solved this problem. We show however that the previous voting scheme was not ballot-private, which is weaker than receipt-freeness.

To the best of our knowledge, this is the first scheme that is both receipt-free (in a strong sense) and has universal verifiability (in the sense of strong verifiability [CGGI14]), without requiring the existence of an untappable channel, or the use of secure hardware tokens. We only require that the receipt-freeness adversary is not eavesdropping the communication between the voter and the voting server, and the existence of a re-randomization service. As a result, we overcome the impossibility result [CFP$^+$10], stating no scheme can be receipt-free and universally verifiable without an untappable channel (this is for instance the case of JCJ/Civitas [JCJ05]). We achieve this by relying on a ballot box server that is charged with re-randomising ballots, and at the same time precluded from changing their contents, in a publicly verifiable way [BFPV11a]. Finally, we showed the feasibility of our approach by implementing a voting client in Javascript and measuring its performance in a number of platforms.

Chapter 10

Conclusions

Throughout this work we examined protocols that aim to enforce correct behaviour between (perhaps untrusting) participants. In some examples this is direct, e.g. in the case of preventing a malicious voter from posting a stuffed ballot. In the case of multi-user signatures this enforcement is more indirect, and lies with the potential for anonymity to be revoked (of course, in the context of revoking anonymity, direct enforcement is again necessary).

None the less, most of the "targets" of the presented research have a history of movement. Efficiency is always a priority in practical applications, and can often enable new applications by itself (e.g highly efficient verification enables the outsourcing of computations [PHGR13]). The notion of security is also evolving, not only because attacks can become more sophisticated, but also because we discover gaps in our modelling or our assumptions. With that in mind, some future directions seem promising in particular:

The Protocol of Chapter 5 is highly efficient, and achieves impressively small proof sizes given its setting. At the same time, the verification time against pairing-based systems such as [PHGR13] is much higher. It would be interesting to see if the iterative reduction technique used can be exported to other applications in the same setting.

The DFN-based construction of Chapter 4 removes the reliance on complexity leveraging and demonstrates that culpable soundness can be justifiably used in some applications. However, it relies on the somewhat inefficient Okamoto-Uchiyama [OU98] cryptosystem, where Paillier [Pai99] or a variant such as [DJN10] might prove more efficient. Additionally, it only covers a somewhat restricted range of protocols. As such, generalizing it to be more expressive is a natural direction. In the same chapter, the delayed message board breaks convention with most current approaches, so it would also be worthwhile to examine whether this requirement can be removed, or alternatively harnessed independently as a potential avenue for improvements: it might be possible to omit revealing contents message board at all by forcing the board to complete a proof certifying they are correct.

A link between multi-user signatures and voting exists by means of list signatures [CSST06], which are akin to group signatures where each user is limited in the number of times they can sign. It would be interesting to see if the efficient accountable ring signature scheme of chapter 6 can be used to produce a list signature apart from a group one (Chap. 7). At the same time, the ring-based construction would also be amenable to self-organizing communities in a similar fashion to self-tallying voting systems [KY02].

BeleniosRF, presented in Chapter 9 obtains good security properties at a low conceptual cost for the voter. While its prover efficiency is acceptable, optimizations might help make it practical for mobile devices sooner rather than later. Being mobile-friendly is more than a matter of convenience for BeleniosRF: as revoting is not currently supported, voting early increases security since users who have voted cannot be coerced. Alternatively, if full coercion resistance is implemented, mobile usage might be made more difficult as coercion-resistant protocols are typically heavier. Additionally, as BeleniosRF gives good solution to the issue of receipts in Helios-like systems (on which the copying system of Chapter 8 relies), it would be interesting to re-examine copying as a potential planned feature of a voting system (rather than unplanned, but technically feasible). As such it could be compared to alternative voting and tallying systems such as STV [BO91].

Bibliography

[AC11] Jordi Puiggalí Allepuz and Sandra Guasch Castelló. Internet voting system with cast as intended verification. In *VoteID'11*. Springer, 2011.

[ACHdM05] Giuseppe Ateniese, Jan Camenisch, Susan Hohenberger, and Breno de Medeiros. Practical group signatures without random oracles. *IACR Cryptology ePrint Archive*, 2005.

[ACJT00] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology – CRYPTO 2000*, pages 255–270, 2000.

[Adi08] Ben Adida. Helios: Web-based Open-Audit Voting. In *USENIX Security Symposium 2008*, pages 335–348, 2008.

[AdMPQ09] Ben Adida, Olivier de Marneffe, Olivier Pereira, and Jean-Jacques Quisquater. Electing a university president using open-audit voting: Analysis of real-world use of Helios. In *EVT/WOTE 2009*, 2009.

[AHO10] Masayuki Abe, Kristiyan Haralambiev, and Miyako Ohkubo. Signing on elements in bilinear groups for modular protocol design. *IACR Cryptology ePrint Archive*, 2010.

[Arr63] Kenneth J. Arrow. *Social Choice and Individual Values*. Yale University Press, New Haven, 2nd edition, 1963.

[AST01] Giuseppe Ateniese, Dawn Song, and Gene Tsudik. Quasi-efficient revocation of group signatures. *IACR Cryptology ePrint Archive*, 2001:101, 2001.

[BBM00] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *Advances in Cryptology – EUROCRYPT 2000*. Springer, 2000.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Advances in Cryptology – CRYPTO 2004*, pages 41–55, 2004.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

[BCC04] Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *Conference on Computer and Communications Security - CCS 2004*, pages 132–145. ACM, 2004.

[BCC$^+$15] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short accountable ring signatures based on DDH. In *European Symposium on Research in Computer Security – ESORICS 2015*, 2015.

[BCC$^+$16a] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. Foundations of fully dynamic group signatures. In *Conference on Applied Cryptography and Network Security – ACNS 2016*, pages 117–136. Springer, 2016.

[BCC$^+$16b] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *Advances in Cryptology – EUROCRYPT 2016*, pages 327–357. Springer, 2016.

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

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In

Symposium on Theory of Computing Conference – STOC 2013, pages 111–120, 2013.

[BCG$^+$13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: verifying program executions succinctly and in Zero Knowledge. In *Advances in Cryptology – CRYPTO 2013*, pages 90–108, 2013.

[BCG$^+$15] David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi. A comprehensive analysis of game-based ballot privacy definitions. In *IEEE Security and Privacy 2015*. IEEE Computer Society, 2015.

[BCKL08] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and noninteractive anonymous credentials. In *Theory of Cryptography Conference – TCC 2008*, pages 356–374. Springer, 2008.

[BCN$^+$10] Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Get shorty via group signatures without encryption. In *Security and Cryptography for Networks - SCN 2010*, pages 381–398, 2010.

[BCNP04] Boaz Barak, Ran Canetti, Jesper Buus Nielsen, and Rafael Pass. Universally composable protocols with relaxed set-up assumptions. In *Foundations of Computer Science – FOCS 2004*, FOCS '04, pages 186–195. IEEE, 2004.

[BCP$^+$11] David Bernhard, Véronique Cortier, Olivier Pereira, Ben Smyth, and Bogdan Warinschi. Adapting Helios for provable ballot privacy. In *European Symposium on Research in Computer Security – ESORICS 2011*, pages 335–354. Springer, 2011.

[BCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *USENIX Security Symposium 2014*, pages 781–796, 2014.

[bel16] BeleniosRF –Voting Client Core. `https://gist.github.com/pyrros/4fddd7d49ae7c9c935f5d6a9a27d14c3`, 2016.

[Ben07] Josh Benaloh. Ballot casting assurance via voter-initiated poll station auditing. In *EVT/WOTE 2007*, 2007.

[BFI$^+$10] Olivier Blazy, Georg Fuchsbauer, Malika Izabachene, Amandine Jambert, Hervé Sibert, and Damien Vergnaud. Batch groth–sahai. In *Conference on Applied Cryptography and Network Security – ACNS 2010*. Springer, 2010.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Symposium on Theory of Computing Conference – STOC '88*, pages 103–112. ACM, 1988.

[BFP$^+$01] Olivier Baudron, Pierre-Alain Fouque, David Pointcheval, Jacques Stern, and Guillaume Poupard. Practical multi-candidate election system. In *Symposium on Principles of distributed computing – PODC 2001*, pages 274–283. ACM, 2001.

[BFPV11a] Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. In *Public Key Cryptography – PKC 2011*, pages 403–422. Springer, 2011.

[BFPV11b] Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. In *Public Key Cryptography - PKC 2011*. Springer, 2011.

[BG11] Katharina Bräunlich and Rüdiger Grimm. Formalization of receipt-freeness in the context of electronic voting. In *Availability, Reliability and Security 2011*. IEEE Computer Society, 2011.

[BG12] Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In *Advances in Cryptology – EUROCRYPT 2012*, pages 263–280, 2012.

[BG13] Stephanie Bayer and Jens Groth. Zero-knowledge argument for polynomial evaluation with application to blacklists. In *Advances in Cryptology – EUROCRYPT 2013*, pages 646–663, 2013.

[BGDM$^+$10] Jean-Luc Beuchat, Jorge E González-Díaz, Shigeo Mitsunari, Eiji Okamoto, Francisco Rodríguez-Henríquez, and Tadanori Teruya. High-speed software implementation of the optimal ate pairing over barreto–naehrig curves. In *Pairing-Based Cryptography – Pairing 2010*. Springer, 2010.

[BGP11] Philippe Bulens, Damien Giry, and Olivier Pereira. Running mixnet-based elections with Helios. In *Electronic Voting Technology Workshop/Workshop on Trustworthy Elections – EVT/WOTE 2011*, 2011.

[BKM09] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. *Journal of Cryptology*, 22(1):114–138, 2009.

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4), 2004.

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Advances in Cryptology – EUROCRYPT 2003*, pages 614–629, 2003.

[BO91] John J Bartholdi and James B Orlin. Single transferable vote resists strategic voting. *Social Choice and Welfare*, 8(4):341–354, 1991.

[BPW12] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In *Advances in Cryptology – ASIACRYPT 2012*, pages 626–643. Springer, 2012.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Conference on Computer and Communications Security - CCS '93*, pages 62–73. ACM, 1993.

[Bri04] E. Brickell. An efficient protocol for anonymously providing assurance of the container of a private key. *Submitted to the Trusted Computing Group*, 2004.

[BS01] Emmanuel Bresson and Jacques Stern. Efficient revocation in group signatures. In *Public Key Cryptography - PKC 2001*, pages 190–206, 2001.

[BS04] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In *Conference on Computer and Communications Security - CCS 2004*, pages 168–177. ACM, 2004.

[BSZ05] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–153. Springer, 2005.

[BT94] Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Symposium on Theory of Computing Conference – STOC '94*, pages 544–553. ACM, 1994.

[BW06] Xavier Boyen and Brent Waters. Compact group signatures without random oracles. In *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 427–444, 2006.

[BW07] Xavier Boyen and Brent Waters. Full-domain subgroup hiding and constant-size group signatures. In *Public Key Cryptography - PKC 2007*, pages 1–15. Springer, 2007.

[BY86] Josh C Benaloh and Moti Yung. Distributing the power of a government to enhance the privacy of voters. In *Symposium on Principles of distributed computing – PODC 1986*, pages 52–62. ACM, 1986.

[Cam97] Jan Camenisch. Efficient and generalized group signatures. In *Advances in Cryptology – EUROCRYPT '97*, volume 1233 of *LNCS*, pages 465–479. Springer, 1997.

[Can89] David G Cantor. On arithmetical algorithms over finite fields. *Journal of Combinatorial Theory, Series A*, 50(2):285–300, 1989.

[CCFG16] Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David Galindo. Beleniosrf: A non-interactive receipt-free electronic voting scheme. In *Conference on Computer and Communications Security - CCS 2016*, pages 1614–1625. ACM, 2016.

[CCM08] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: Toward a secure voting system. In *IEEE Security and Privacy 2008*. IEEE Computer Society, 2008.

[CCS08] Jan Camenisch, Rafik Chaabouni, and Abhi Shelat. Efficient protocols for set membership and range proofs. In *Advances in Cryptology – ASIACRYPT 2008*, pages 234–252. Springer, 2008.

[CD98] Ronald Cramer and Ivan Damgård. Zero-knowledge proofs for finite field arithmetic; or: Can zero-knowledge be for free? In *Advances in Cryptology – CRYPTO 1998*, pages 424–441, 1998.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology – CRYPTO'94*, pages 174–187. Springer, 1994.

[CEC$^+$08] David Chaum, Aleks Essex, Richard Carback, Jeremy Clark, Stefan Popoveniuc, Alan Sherman, and Poorvi Vora. Scantegrity: end-to-end voter-verifiable optical-scan voting. *IEEE Security and Privacy*, 6(3), 2008.

[Cer15] CertiVox. A Cryptographic Library for the Internet of Things. `https://github.com/CertiVox/MiotCL`, 2015.

[CF85] JD Cohen and MJ Fisher. A robust and verifiable cryptographically secure election system. In *Foundations of Computer Science – FOCS '85*, pages 372–382, 1985.

[CFP$^+$10] Benoît Chevallier-Mames, Pierre-Alain Fouque, David Pointcheval, Julien Stern, and Jacques Traoré. On some incompatible properties of voting schemes. In *EVT/WOTE 2010*, 2010.

[CFSY96] Ronald Cramer, Matthew Franklin, Berry Schoenmakers, and Moti Yung. Multi-authority secret-ballot elections with linear work. In *Advances in Cryptology – EUROCRYPT'96*, pages 72–83. Springer, 1996.

[CG96] Ran Canetti and Rosario Gennaro. Incoercible multiparty computation (extended abstract). In *Foundations of Computer Science – FOCS '96*. IEEE Computer Society, 1996.

[CG05] Jan Camenisch and Jens Groth. Group signatures: Better efficiency and new theoretical aspects. In *Security in Communication Networks - SCN 2004*, pages 120–133, 2005.

[CG15] Pyrros Chaidos and Jens Groth. Making sigma-protocols non-interactive without random oracles. In *Public Key Cryptography – PKC 2015*, pages 650–670. Springer, 2015.

[CGGI13] Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachène. Distributed ElGamal á la Pedersen: Application to Helios. In *Privacy in the Electronic Society*, WPES '13, pages 131–142. ACM, 2013.

[CGGI14] Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachène. Election verifiability for Helios under weaker trust assumptions. In *European Symposium on Research in Computer Security – ESORICS 2014*. Springer, 2014.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *Journal of the ACM (JACM)*, 51(4):557–594, 2004.

[CGS97a] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. *European transactions on Telecommunications*, 8(5):481–490, 1997.

[CGS97b] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology – EUROCRYPT '97*. Springer, 1997.

[Cha81] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, February 1981.

[Cha83] D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology – CRYPTO '83*, volume 82, pages 199–203, 1983.

[CKN03] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. Relaxing chosen-ciphertext security. In *Advances in Cryptology – CRYPTO 2003*. Springer, 2003.

[CKS09] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In *Public Key Cryptography - PKC 2009*, pages 481–500. Springer, 2009.

[CL02] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology – CRYPTO 2002*, pages 61–76, 2002.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology – CRYPTO 2004*, pages 56–72, 2004.

[CL06] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, 2006.

[CLW08] Sherman S. M. Chow, Joseph K. Liu, and Duncan S. Wong. Robust receipt-free election system with ballot secrecy and verifiability. In *Network and Distributed System Security Symposium 2008*. The Internet Society, 2008.

[CM98] Jan Camenisch and Markus Michels. A group signature scheme with improved efficiency. In *Advances in Cryptology – ASIACRYPT '98*, pages 160–174. Springer, 1998.

[CMFPT06] B. Chevallier-Mames, P.A. Fouque, D. Pointcheval, and J. Traoré. On some incompatible properties of voting schemes. In *Workshop On Trustworthy Elections – WOTE'06*. Citeseer, 2006.

[CP93] D. Chaum and T. Pedersen. Wallet databases with observers. In *Advances in Cryptology – CRYPTO'92*, pages 89–105. Springer, 1993.

[Cro06] D. Crockford. Javascript object notation. http://www.ietf.org/rfc/rfc4627.txt, July 2006.

[CS97] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In *Advances in Cryptology – CRYPTO '97*, pages 410–424, 1997.

[CS98] Ronald Cramer and Victor Shoup. A practical public-key encryption schemes secure against adaptive chosen ciphertext attack. In *Advances in Cryptology – CRYPTO'98*, pages 13–25. Springer, 1998.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Advances in Cryptology – EUROCRYPT 2002*, pages 45–64. Springer, 2002.

[CS11] Véronique Cortier and Ben Smyth. Attacking and fixing Helios: An analysis of ballot secrecy. In *Computer Security Foundations Symposium – CSF'11*, 2011.

[CS13] Véronique Cortier and Ben Smyth. Attacking and fixing Helios: An analysis of ballot secrecy. *Journal of Computer Security*, 21(1), 2013.

[CSST06] Sébastien Canard, Berry Schoenmakers, Martijn Stam, and Jacques Traoré. List signature schemes. *Discrete Applied Mathematics*, pages 189 – 201, 2006. Coding and Cryptography.

[CvH91] David Chaum and Eugène van Heyst. Group signatures. In *Advances in Cryptology – EUROCRYPT '91*, pages 257–265, 1991.

[Dam92] Ivan Damgård. Non-interactive circuit based proofs and non-interactive perfect zero-knowledge with preprocessing. In *Advances in Cryptology – EUROCRYPT'92*, pages 341–355. Springer, 1992.

[Dan15] George Danezis. petlib: A python library that implements a number of privacy enhancing technologies (PETs), 2015.

[DC11] Yvo Desmedt and Pyrros Chaidos. Blinding ballot copying in Helios: from Condorcet to IACR. In *Advances in Cryptology – CRYPTO 2011 Rump Session*, 2011.

[DC12] Yvo Desmedt and Pyrros Chaidos. Applying divertibility to blind ballot copying in the helios internet voting system. In *European Symposium on Research in Computer Security – ESORICS 2012*, pages 433–450. Springer, 2012.

[DF90] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In *Advances in Cryptology – CRYPTO'89*, pages 307–315. Springer, 1990.

[DFN06] Ivan Damgård, Nelly Fazio, and Antonio Nicolosi. Non-interactive zero-knowledge from homomorphic encryption. In *Theory of Cryptography Conference – TCC 2006*, pages 41–59. Springer, 2006.

[DGB88] Y. Desmedt, C. Goutier, and S. Bengio. Special uses and abuses of the Fiat-Shamir passport protocol. In *Advances in Cryptology – CRYPTO '87*, pages 21–39, 1988.

[DGS03] Ivan Damgård, Jens Groth, and Gorm Salomonsen. The theory and implementation of an electronic voting system. In *Secure Electronic Voting*, pages 77–99. Springer, 2003.

[DJ03] Ivan Damgård and Mads Jurik. A length-flexible threshold cryptosystem with applications. In *Information Security and Privacy – ACISP 2003*, pages 350–364. Springer, 2003.

[DJN10] Ivan Damgård, Mads Jurik, and Jesper Buus Nielsen. A generalization of Paillier's public-key system with applications to electronic voting. *International Journal of Information Security*, 9(6):371–385, 2010.

[DK05] Y. Desmedt and K. Kurosawa. Electronic voting: Starting over? *Information Security*, pages 329–343, 2005.

[DKNS04] Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in ad hoc groups. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, pages 609–626. Springer, 2004.

[DKR06] Stéphanie Delaune, Steve Kremer, and Mark Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *Computer Security Foundations Workshop – CSFW 2006*. IEEE Computer Society, 2006.

[DP06] Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In *Progressing Cryptology - VIETCRYPT 2006*, pages 193–210, 2006.

[DVDGA12] Denise Demirel, Jeroen Van De Graaf, and Roberto Araújo. Improving helios with everlasting privacy towards the public. *EVT/WOTE*, 2012.

[ED10] Saghar Estehghari and Yvo Desmedt. Exploiting the client vulnerabilities in internet e-voting systems: hacking Helios 2.0 as an example. In *Electronic Voting Technology Workshop/Workshop on Trustworthy Elections – EVT-WOTE '10*, pages 1–9. USENIX Association, 2010.

[Elg85] Taher Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *Information Theory, IEEE Transactions on*, 31(4):469 – 472, Jul 1985.

[Est10] Estonian National Electoral Committee. E-voting system -general overview, 2010.

[Fel87] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Foundations of Computer Science – FOCS '87*, pages 427–438. IEEE, 1987.

[FI05] Jun Furukawa and Hideki Imai. An efficient group signature scheme from bilinear maps. In *ACISP*, pages 455–467, 2005.

[Fis05] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *Advances in Cryptology – CRYPTO 2005*, pages 152–168. Springer, 2005.

[FKMV12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the Fiat-Shamir transform. In *Progress in Cryptology – INDOCRYPT*, pages 60–79. Springer, 2012.

[FP09] Georg Fuchsbauer and David Pointcheval. Proofs on encrypted values in bilinear groups and an application to anonymity of signatures. In *Pairing-Based Cryptography – Pairing 2009*. Springer, 2009.

[FS87] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology – CRYPTO '86*, pages 186–194. Springer, 1987.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *Symposium on Theory of Computing Conference – STOC 1990*, pages 416–426. ACM, 1990.

[FS08] Eiichiro Fujisaki and Koutarou Suzuki. Traceable ring signature. *IEICE Transactions*, 91-A(1):83–93, 2008.

[FY04] Jun Furukawa and Shoko Yonezawa. Group signatures with separate and distributed authorities. In *Security in Communication Networks – SCN 2004*, pages 77–90, 2004.

[FZ13] Matthew K. Franklin and Haibin Zhang. Unique ring signatures: A practical construction. In *Financial Cryptography and Data Security – FC 2013*, pages 162–170. Springer, 2013.

[GCG15] Stéphane Glondu, Véronique Cortier, and Pierrick Gaudry. Belenios – Verifiable online voting system. http://belenios.gforge.inria.fr, 2015.

[GGI$^+$14] Craig Gentry, Jens Groth, Yuval Ishai, Chris Peikert, Amit Sahai, and Adam Smith. Using fully homomorphic hybrid encryption to minimize non-interative zero-knowledge proofs. *Journal of Cryptology*, pages 1–24, 2014.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In *Advances in Cryptology – EUROCRYPT 2013*, pages 626–645. Springer, 2013.

[GGR09] Ryan W. Gardner, Sujata Garera, and Aviel D. Rubin. Coercion resistant end-to-end voting. In *Financial Cryptography and Data Security – FC 2009*. Springer, 2009.

[GH98] Oded Goldreich and Johan Håstad. On the complexity of interactive proofs with bounded communication. *Information Processing Letters*, 67(4):205–214, 1998.

[GHR15] Alonso González, Alejandro Hevia, and Carla Ràfols. Qa-nizk arguments in asymmetric groups: new tools and new constructions. In *Advances in Cryptology – ASIACRYPT 2015*. Springer, 2015.

[GI08] Jens Groth and Yuval Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In *Advances in Cryptology – EUROCRYPT 2008*, pages 379–396. 2008.

[Gjø12] Kristian Gjøsteen. The norwegian internet voting protocol. In *VoteID'11*, pages 1–18, 2012.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *Foundations of Computer Science – FOCS 2003*, FOCS '03, pages 102–113. IEEE, 2003.

[GK14] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *Advances in Cryptology – EUROCRYPT 2015*, page 764, 2014.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991.

[GMY06] Juan a. Garay, Philip MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. *Journal of Cryptology*, 19(2):169–209, 2006.

[Gol11] Oded Goldreich. *Studies in Complexity and Cryptography*, chapter Basing Non-interactive Zero-knowledge on (Enhanced) Trapdoor Permutations: The State of the Art, pages 406–421. Springer, 2011.

[GOM16] Irene Giacomelli, Claudio Orlandi, and Jesper Madsen. ZKBoo: Faster zero-knowledge for boolean circuits. In *USENIX Security Symposium 2016*, pages 1069–1083. USENIX Association, 2016.

[GOS12] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *Journal of the ACM*, 59(3):11:1–11:35, 2012.

[GQ88] Louis C. Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both trasmission and memory. In *Advances in Cryptology – EUROCRYPT 1998*, pages 123–128, 1988.

[Gro04a] Jens Groth. Efficient maximal privacy in boardroom voting and anonymous broadcast. In *Financial Cryptography and Data Security – FC 2004*, pages 90–104. Springer, 2004.

[Gro04b] Jens Groth. *Honest verifier zero-knowledge arguments applied.* PhD thesis, University of Aarhus, 2004.

[Gro06] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *Advances in Cryptology – ASIACRYPT 2006*, pages 444–459. Springer, 2006.

[Gro07] Jens Groth. Fully anonymous group signatures without random oracles. In *Advances in Cryptology – ASIACRYPT 2007*, pages 164–180, 2007.

[Gro09a] Jens Groth. Efficient zero-knowledge arguments from two-tiered homomorphic commitments. In *Advances in Cryptology – ASIACRYPT 2009*, pages 431–448, 2009.

[Gro09b] Jens Groth. Linear algebra with sub-linear zero-knowledge arguments. In *Advances in Cryptology – CRYPTO 2009*, pages 192–208, 2009.

[Gro10a] Jens Groth. Short non-interactive zero-knowledge proofs. In *Advances in Cryptology – ASIACRYPT 2010*, pages 341–358. Springer, 2010.

[Gro10b] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *Advances in Cryptology – ASIACRYPT 2010*, pages 321–340. Springer, 2010.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *Advances in Cryptology – EUROCRYPT 2008*. Springer, 2008.

[GS12] Jens Groth and Amit Sahai. Efficient noninteractive proof systems for bilinear groups. *SIAM Journal on Computing*, 41(5):1193–1232, 2012.

[GVW02] Oded Goldreich, Salil Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. *Computational Complexity*, 11(1-2):1–53, 2002.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Symposium on Theory of Computing Conference – STOC 2011*, pages 99–108. ACM, 2011.

[Hel11] Helios Voting. Helios v3 verification specs. http://documentation.heliosvoting.org/verification-specs/helios-v3-verification-specs, August 2011.

[Hir10] Martin Hirt. Receipt-free K-out-of-L voting based on ElGamal encryption. In *EVT/WOTE 2010*. Springer, 2010.

[HK11] F Hao and MN Kreeger. Every vote counts: ensuring integrity in dre-based voting system. Technical report, Technical Report, 2011.

[HN06] Martin Hirt and Jesper Buus Nielsen. Robust multiparty computation with linear communication complexity. In *Advances in Cryptology – CRYPTO 2006*, CRYPTO'06, pages 463–482. Springer, 2006.

[HS00] Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. In *Advances in Cryptology – EUROCRYPT 2000*. Springer, 2000.

[HS12] James Heather and Steve Schneider. A formal framework for modelling coercion resistance and receipt freeness. In *Formal Methods 2012*. Springer, 2012.

[IAC] International Association for Cryptologic Research, elections. Page at http://www.iacr.org/elections/.

[JCJ05] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *Workshop on Privacy in the Electronic Society 2005*. ACM, 2005.

[JdV06] Hugo L. Jonker and Erik P. de Vink. Formalising receipt-freeness. In *Information Security 2006*. Springer, 2006.

[JSI96] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In *Advances in Cryptology – EUROCRYPT'96*, pages 143–154. Springer, 1996.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *Symposium on Theory of Computing Conference – STOC 1992*, pages 723–732, 1992.

[KP98] Joe Kilian and Erez Petrank. An efficient noninteractive zero-knowledge proof system for NP with general assumptions. *Journal of Cryptology*, 11(1):1–27, 1998.

[KT09] Ralf Küsters and Tomasz Truderung. An epistemic approach to coercion-resistance for electronic voting protocols. In *Security and Privacy – S&P 2009*. IEEE Computer Society, 2009.

[KTV15] Oksana Kulyk, Vanessa Teague, and Melanie Volkamer. Extending Helios towards private eligibility verifiability. In *E-Voting and Identity 2015*. Springer, 2015.

[KTY04] Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Traceable signatures. In *Advances in Cryptology – EUROCRYPT 2004*, pages 571–589, 2004.

[KY02] Aggelos Kiayias and Moti Yung. Self-tallying elections and perfect ballot secrecy. In *Public Key Cryptography – PKC 2002*, pages 141–158. Springer, 2002.

[KY05] Aggelos Kiayias and Moti Yung. Group signatures with efficient concurrent join. In *Advances in Cryptology – EUROCRYPT 2005*, pages 198–214. Springer, 2005.

[KY06] Aggelos Kiayias and Moti Yung. Secure scalable group signature with dynamic joins and separable authorities. *IJSN*, 1(1/2):24–45, 2006.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In *Advances in Cryptology – ASIACRYPT 2010*, pages 177–194, 2010.

[KZZ15] Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. End-to-end verifiable elections in the standard model. In *Advances in Cryptology – EUROCRYPT 2015*. Springer, 2015.

[Len05] Arjen K Lenstra. *Key lengths*. Wiley, 2005.

[Lim00] Chae Hoon Lim. Efficient multi-exponentiation and application to batch verification of digital signatures, 2000. Manuscript available at `http://dasan.sejong.ac.kr/~chlim/pub/multi_exp.ps`.

[Lin03] Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. *Journal of Cryptology*, 16(3):143–184, 2003.

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *Theory of Cryptography Conference – TCC 2012*, pages 169–189, 2012.

[LLNW14] Adeline Langlois, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-based group signature scheme with verifier-local revocation. In *Public-Key Cryptography – PKC 2014*, pages 345–361, 2014.

[LPY12a] Benoît Libert, Thomas Peters, and Moti Yung. Group signatures with almost-for-free revocation. In *Advances in Cryptology – CRYPTO 2012*, pages 571–589, 2012.

[LPY12b] Benoît Libert, Thomas Peters, and Moti Yung. Scalable group signatures with revocation. In *Advances in Cryptology – EUROCRYPT 2012*, pages 609–627, 2012.

[LV09] Benoît Libert and Damien Vergnaud. Group signatures with verifier-local revocation and backward unlinkability in the standard model. In *Cryptology and Network Security - CANS*, pages 498–517, 2009.

[LWW04] Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups. In *ACISP*, volume 3108, pages 325–335, 2004.

[MBC01] Emmanouil Magkos, Mike Burmester, and Vassilios Chrissikopoulos. Receipt-freeness in large-scale elections without untappable channels. In *E-Commerce, E-Business, E-Government 2001*. Kluwer, 2001.

[Mic94] Silvio Micali. Cs proofs. In *Foundations of Computer Science – FOCS '94*, pages 436–453. IEEE, 1994.

[MN06] Tal Moran and Moni Naor. Receipt-free universally-verifiable voting with everlasting privacy. In *Advances in Cryptology – CRYPTO 2006*. Springer, 2006.

[Möl01] Bodo Möller. Algorithms for multi-exponentiation. In *Selected Areas in Cryptography – SAC 2001*, pages 165–180. Springer, 2001.

[MR08] Bodo Möller and Andy Rupp. Faster multi-exponentiation through caching: accelerating (EC) DSA signature verification. In *Security and Cryptography for Networks – SCN 2008*, pages 39–56. Springer, 2008.

[NF05] Toru Nakanishi and Nobuo Funabiki. Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps. In *Advances in Cryptology – ASIACRYPT 2005*, pages 533–548, 2005.

[NFHF09] Toru Nakanishi, Hiroki Fujii, Yuta Hira, and Nobuo Funabiki. Revocable group signature schemes with constant costs for signing and verifying. In *Public Key Cryptography - PKC 2009*, pages 463–480, 2009.

[Ngu05] Lan Nguyen. Accumulators from bilinear pairings and applications. In *CT-RSA 2005*, pages 275–292. Springer, 2005.

[NNL01] Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In *Advances in Cryptology – CRYPTO 2001*, pages 41–62, 2001.

[NS04] Lan Nguyen and Reihaneh Safavi-Naini. Efficient and provably secure trapdoor-free group signature schemes from bilinear pairings. In *Advances in Cryptology – ASIACRYPT 2004*, pages 372–386, 2004.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Symposium on Theory of Computing Conference – STOC 1990*, STOC 2013, pages 427–437. ACM, 1990.

[Oka97] Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Security Protocols 97*. Springer, 1997.

[Oli06] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[OO90] Tatsuaki Okamoto and Kazuo Ohta. Divertible zero knowledge interactive proofs and commutative random self-reducibility. In *Advances in Cryptology – EUROCRYPT '98*, pages 134–149. Springer, 1990.

[OU98] Tatsuaki Okamoto and Shigenori Uchiyama. A new public-key cryptosystem as secure as factoring. In *Advances in Cryptology – EUROCRYPT'98*, pages 308–318. Springer, 1998.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – EUROCRYPT'99*, pages 223–238. Springer, 1999.

[Ped91] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – CRYPTO '91*, volume 576 of *LNCS*, pages 129–140. Springer, 1991.

[Pen46] Lionel Sharples Penrose. The elementary statistics of majority voting. *Journal of the Royal Statistical Society*, 109(1):53–57, 1946.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *Security and Privacy – S&P 2013*, pages 238–252. IEEE, 2013.

[RBH$^+$09] Peter YA Ryan, David Bismark, James A Heather, Steve A Schneider, and Zhe Xia. The Prêt à Voter verifiable election system. *IEEE Transactions on Information Forensics and Security*, 4, 2009.

[RRI15] Peter Y A Ryan, Peter B Roenne, and Vincenzo Iovino. Selene: Voting with transparent verifiability and coercion-mitigation. Cryptology ePrint Archive, Report 2015/1105, 2015. http://eprint.iacr.org/.

[RS91] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology – CRYPTO '91*, pages 433–444. Springer, 1991.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *Advances in Cryptology – ASIACRYPT 2001*, pages 552–565. Springer, 2001.

[SB13] Ben Smyth and David Bernhard. Ballot secrecy and ballot independence coincide. In *European Symposium on Research in Computer Security – ESORICS 2013*, pages 463–480. Springer, 2013.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[Seo11] Jae Hong Seo. Round-efficient sub-linear zero-knowledge arguments for linear algebra. In *Public Key Cryptography – PKC 2011*, pages 387–402, 2011.

[SFD$^+$14] Drew Springall, Travis Finkenauer, Zakir Durumeric, Jason Kitcat, Harri Hursti, Margaret MacAlpine, and J. Alex Halderman. Security analysis of the estonian internet voting system. In *Conference on Computer and Communications Security - CCS 2014*. ACM, 2014.

[Sho01] Victor Shoup. NTL: A library for doing number theory, 2001.

[SK95] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme. In *Advances in Cryptology – EUROCRYPT'95*, pages 393–403. Springer, 1995.

[Son01] Dawn Xiaodong Song. Practical forward secure group signature schemes. In *Conference on Computer and Communications Security - CCS 2001*, pages 225–234. ACM, 2001.

[SSE$^+$12] Yusuke Sakai, Jacob C. N. Schuldt, Keita Emura, Goichiro Hanaoka, and Kazuo Ohta. On the security of dynamic group signatures: Preventing signature hijacking. In *Public Key Cryptography - PKC 2012*, pages 715–732, 2012.

[TS06] Isamu Teranishi and Kazue Sako. k-times anonymous authentication with a constant proving cost. In *Public Key Cryptography - PKC 2006*, pages 525–542, 2006.

[TX03] Gene Tsudik and Shouhuai Xu. Accumulating composites and improved group signing. In *Advances in Cryptology – ASIACRYPT 2003*, pages 269–286, 2003.

[VV09] Carmine Ventre and Ivan Visconti. Co-sound zero-knowledge with public keys. In *Progress in Cryptology–AFRICACRYPT 2009*, pages 287–304. Springer, 2009.

[Wat05] Brent R. Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology – EUROCRYPT 2005*. Springer, 2005.

[Wik08] Douglas Wikström. Simplified submission of inputs to protocols. In *Security and Cryptography for Networks – SCN 2008*, pages 293–308. Springer, 2008.

[XY04] Shouhuai Xu and Moti Yung. Accountable ring signatures: A smart card approach. In *CARDIS*, volume 153 of *IFIP*, pages 271–286. Springer, 2004.