Non-interactive arguments of knowledge

Michele Orrù

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à l'École Normale Supérieure

Non-interactive arguments of knowledge

Soutenue par

Michele ORRÙ

Le 7 April 2020

École doctorale n°386

Sciences Mathématiques de Paris Centre

Spécialité

Informatique

Composition du jury :

David Pointcheval	CNRS, École Normale Supérieure	*Président*
Sarah Meiklejohn	University College London	*Rapporteur*
Moti Yung	Columbia, Google	*Rapporteur*
Pierre-Alain Fouque	Université Rennes 1	*Examinateur*
Benoît Libert	CNRS, École Normale Supérieure de Lyon	*Examinateur*
Georg Fuchsbauer	Inria, École Normale Supérieure	*Directeur de thèse*
Hoeteck Wee	CNRS, École Normale Supérieure	*Codirecteur de thèse*

1794
ENS | PSL

Non-interactive arguments of knowledge

Michele ORRÙ

7 April 2020

Abstract

In this manuscript, we study non-interactive arguments of knowledge, a cryptographic primitive that allows a prover to convince a verifier of the truth of a certain statement. In particular, we will analyze cryptographic constructions that allow a user to prove knowledge of a so-called witness x that satisfies a circuit $\mathtt{C}$, i.e. such that $\mathtt{C}(x) = 1$. We will focus on protocols that hide x while simultaneously guaranteeing soundness of the system. That is, cryptographic schemes that make it hard for the verifier to learn information about the input x that satisfies $\mathtt{C}$.

First, we prove the existence of witness-indistinguishable non-interactive arguments of knowledge in the standard model. We call these protocols non-interactive zaps of knowledge, or zaks. Secondly, we revisit a family of zero-knowledge arguments of knowledge (SNARKs) that is particularly appealing for real-world applications due to its short (constant) proof size. We show that it can be moved to post-quantum assumptions, as long as the verifier is known in advance. We provide an implementation and extended benchmarks for this construction. Lastly, we consider a novel, anonymous cryptocurrency whose security can be guaranteed via arguments of knowledge: Mimblewimble. The cryptocurrency was proposed by an anonymous author in 2016. We provide the first formal analysis of it, fixing a security issue present in the initial proposal.

Acknowledgments

Je tiens d'abord à remercier Georg Fuchsbauer pour avoir encadré ma recherche au long de mon doctorat. Merci de m'avoir accueilli, merci de ta grande patience avec moi, et d'avoir pris le temps de m'expliquer la cryptographie.

Je suis très reconnaissant à Sarah Meiklejohn et Moti Yung d'avoir accepté d'être les rapporteur.e.s de ma thèse en sacrifiant une partie de leur temps. Je remercie aussi Pierre-Alain Fouque et Benoît Libert de me faire l'honneur de participer au jury de soutenance, ainsi que David Pointcheval, pour avoir été disponible depuis le début de ma recherche de thèse, jusqu'à sa toute fin, en répondant à mes questions de nature bureaucratique, technique et ludique.

Mes remerciements vont aussi à tou.s.tes mes collègues de l'équipe Crypto à l'École Normale Supérieure, présent.e.s et désormais vieu.x.illes. Merci pour les cafés et les biscuits partagés. Merci d'avoir supporté mes idées, merci d'avoir réfléchi avec moi, et de m'avoir adopté <3.

Également, je suis reconnaissant envers les personnes qui composent l'équipe administrative de l'ENS ; merci à Lise-Marie au DI, Christian au théâtre, Virginia au Pôt, et Manu à la loge pour avoir rendu mon environnement de travail si apaisant, même dans des conditions de travail difficiles.

J'aimerais également exprimer ma gratitude à tou.te.s mes coaut.eur.ice.s : sans vous je n'aurais aucun résultat à présenter aujourd'hui. Merci Googlers, de m'avoir permis de participer à une chouette expérience.

Je voudrais remercier aussi celleux qui ont financé mes recherches et mes déplacements, en particulier: Hoeteck Wee et le CNRS, pour avoir couvert la plupart de mes frais et de mon salaire. Zaki Manian, pour m'avoir offert un logement en conférence quand je ne savais pas comment le financer. Jeff Brudges, pour m'avoir offert un parachute entre ma thèse et ma future carrière.

Pour conclure, je voudrais remercier celleux qui m'ont accompagné le plus pendant toutes ces années : le Chaos Computer Club, pour m'avoir appris la différence entre nerd et hacker ; les Torroristes, pour m'avoir appris la différence entre pouvoir et autorité ; les humaines de MirageOS, pour m'avoir appris les affinités entre oeuvre d'art et quelques lignes de code.

Le dernier mot est pour celleux qui me sont les plus ch.ers.ères, l'équipe K-Fêt et plus en général la vie associative de l'École. Merci d'être ma famille. Merci à `C8E` pour avoir été toujours présent, depuis mes 13 ans, à quelques hops de distance.

$-\mu$.

Contents

Chapter 1

Introduction

Cryptography is the science of secure communication. It configures who can do what, from what [Rog15]. In many applications today, securing the communication means allowing users to prove that they are acting according to a given protocol and yet only reveal information that is necessary [Cha08]. Zero-knowledge (ZK) proof systems [GMR89] allow us to prove statements about user secrets, without giving away anything more than what is strictly necessary. They can be used to prove that we are who we say we are, while never meeting in person, and while giving away nothing that could be used to impersonate us in future [Cha08]. The essence of zero-knowledge proof systems is that it's possible to *prove* the truth of a certain statement and successfuly convince another party, without revealing any additional information besides the truth of the statement. Soundness of a proof system demands that if the proof verifies, then the statement is true.

A natural strengthening of security for a proof systems if a so-called proof of knowledge: it guarantees that, whenever the verifier is convinced by an efficient prover, not only the statement was true, but also the prover had *knowledge* of a *witness* associated to the statement. This fact is captured by requiring the existence of a knowledge extractor that can extract a valid witness from the proof. This stronger notion is essential in certain applications of zero-knowledge proofs such as anonymous credentials (cf. Section 1.4) and confidential transactions (cf. Chapter 5).

In this thesis, we are concerned with the design of cryptographic protocols (or cryptosystems) on the topic of non-interactive proofs of knowledge. We restrict ourselves to proofs of knowledge where security holds computationally, also called arguments of knowledge. Settling on computational security allows for significant savings in communication (and verification) [Wee07], leading to schemes that can be implemented in practice. We focus on arguments of knowledge for which we can guarantee strong privacy guarantees: when ZK cannot be achieved, we focus on the (weaker) notion of witness indistinguishability.

1.1 Proof systems

Proof systems are interactive protocols where a user (called *the prover*) attempts to convince another user (called *the verifier*) that a certain proposition is true. We say that a proof system is *non-interactive* if the prover sends a single message to deliver the whole "proof". In a proof system, both parties are formalized as Turing Machines; the prover is computationally unbounded, while the verifier's runtime is polynomially bounded. The protocol is such that it's not possible to convince the verifier of a wrong statement. This property is called *soundness*, or validity.

P versus NP. Proof systems are not only a cryptographic tool but also an object of study in theoretical computer science. One of the most important open problems in computer science is the problem of P versus NP. Essentially, it asks whether every problem whose solution can be

efficiently verified can also be efficiently solved. It is one of the seven Millennium Prize Problems of the Clay Mathematics Institute. More formally, the class P is the class of languages $\mathcal{L}$ that can be decided in polynomial time. NP is the class of languages $\mathcal{L}$ such that there exists an algorithm that takes as input two bit strings x and w and that can decide in polynomial time (in the size of x), whether w is a valid proof or witness that $x \in \mathcal{L}$. We suppose that for any statement x there exists such a witness w, while otherwise, no such witness exists.

The complexity class corresponding to interactive proofs is denoted IP and was studied in two seminal papers by Babai [Bab85], and Goldwasser, Micali, and Rackoff [GMR85]. Interestingly, the power of interactive proof systems is not decreased if the verifier is only allowed random queries (that is, if it merely tosses coins and sends any outcome to the prover) [GS86].

Shamir [Sha90] showed that IP is equal to the class PSPACE, which corresponds to algorithms that use a polynomial amount of space to store intermediate variables. The class PSPACE contains all languages that can be recognized by an algorithm that uses polynomial space, but which is allowed unbounded running time. In particular, the class PSPACE contains the class NP.

Another characterization of NP was provided via Probabilistically Checkable Proofs (PCP). A $\mathsf{PCP}[r(\lambda), q(\lambda)]$ is a type of proof that can be verified in polynomial time using at most $O(r(\lambda))$ random bits and by reading at most $O(q(\lambda))$ bits of the proof. The so-called PCP theorem [ALM+92] states that $\mathsf{NP} = \mathsf{PCP}[\log \lambda, 1]$. In other words, any language in NP can be encoded as a PCP that can be verified by only reading a constant number of bits and by using $O(\log \lambda)$ bits of randomness.

NIZK. An interactive proof system is said to be *zero-knowledge* (ZK) if the proof does not reveal any information besides the truth of the proposition (and whatever can be inferred from it). The concept of zero-knowledge proof systems was first proposed in [GMR89] and is a central tool in modern cryptography. Consider an NP relation R which defines the language of all statements x for which there exists a witness w so that $(x, w) \in \mathsf{R}$. In a zero-knowledge proof for R a prover, knowing a witness, wants to convince a verifier that x is in the language, without revealing any additional information about the witness. More precisely, a ZK proof system must be *complete*, that is, if the prover knows a witness for x then it can convince the verifier; *sound*, in that no malicious prover can convince the verifier of a false statement; and *zero-knowledge*: the execution of the protocol reveals no information to the verifier (beyond the fact that x is in the language). Soundness protects the verifier; zero knowledge protects the prover.

For practical applications, researchers immediately recognized two limiting factors in zero-knowledge proofs: the original protocols were interactive and the proof could be as long as (if not longer than) the witness. *Non-interactive* zero-knowledge proofs (NIZK) proofs [BFM88] and succinct ZK arguments [Kil92, Mic94] were introduced shortly thereafter. NIZK allow the prover to convince the verifier by only sending a single message. However, NIZK must rely on the existence of a common reference string (CRS) to which prover and verifier have access [GO94]. The CRS is assumed to have been set up by some trusted party, which represents a serious limitation for all applications of NIZK in scenarios where parties mutually distrust each other.

Zaps. Feige and Shamir [FS90] proposed a relaxation of zero knowledge called *witness indistinguishability*, which only requires that it is indistinguishable which witness was used to compute a proof. This notion turns out to be sufficient in many contexts. Dwork and Naor [DN00] constructed a two-round witness-indistinguishable proof system for NP in the plain model, that is, where no trusted CRS is assumed. In their protocol, the first message (sent from the verifier to the prover) can be fixed once and for all, and the second one provides the actual proof. They called such protocols *zaps*. Barak, Ong, and Vadhan [BOV03] introduced the concept of *non-interactive* zaps, where the prover sends a single message to deliver the proof. Non-interactive zaps are thus

non-interactive proof systems *without* a CRS. Since in this scenario it is impossible to achieve zero knowledge [GO94], witness indistinguishability (WI) is the best one can hope for.

Groth, Ostrovsky, and Sahai constructed the first non-interactive zaps from standard assumptions [GOS06a]. Subsequently, Ràfols extended this line of research [Ràf15]. All these zaps satisfy soundness, that is, a valid proof can only be computed for valid statements. *Knowledge-soundness* is an alternative notion of validity that not only demands the statement to be true, but also that the prover knows a witness for it. More specifically, knowledge-soundness requires that for each prover there exists an efficient extractor which can extract a witness from the prover whenever the latter outputs a valid proof. (When this holds for computationally bounded provers only, we speak of *arguments* rather than proofs [BCC88].) Since, by definition, false statements have no witnesses, knowledge-soundness implies the standard notion of (computational) soundness.

More recently, Bellare, Fuchsbauer and Scafuro investigated the security of NIZK in the face of parameter subversion and observed that NI zaps provide subversion-resistant soundness and WI. This leaves open the question of whether it is possible to construct a subversion resistant WI zap that is knowledge-sound. We tackle this question in Chapter 3.

SNARKs. Starting from Kilian's protocol [Kil92], Micali [Mic94] constructed a *one-message* succinct argument for NP whose soundness is set in the random oracle model. *Succinctness* here means that the proof length is independent of the witness. Such systems are called *succinct non-interactive arguments* (SNARGs) [GW11]. Several SNARGs constructions have been proposed [Gro10, GGPR13, Gro16], and the area of SNARGs has become popular in the last years with the proposal of constructions which introduced significant improvements in efficiency. An important remark is that all such constructions are based on non-falsifiable assumptions [Nao03], a class of assumptions that is likely to be inherent in proving the security of SNARGs for general NP languages (without random oracles), as shown by Gentry and Wichs [GW11].

Many SNARGs are also *arguments of knowledge* [BLCL91] – so called SNARKs [BCCT12, BCC$^+$14]. That is, they satisfy the stronger notion of validity called knowledge-soundness. SNARKs were initially introduced for verifiable computation and are now the most widely deployed proof systems. For instance, they are used in cryptocurrencies such as Zcash [BCG$^+$14], which guarantees confidentiality of monetary transactions via zero-knowledge SNARKs (zk-SNARKs). Zero-knowledge SNARKs are non-interactive succint arguments of knowledge that satisfy zero knowledge. As for all NIZK systems, a drawback of zk-SNARKs is that they require a CRS, that is, they require a one-time trusted setup of public parameters. Since for SNARKs every CRS has a simulation trapdoor, subversion of these parameters leads to full compromise of soundness [BFS16].

1.2 Instantiations and applications

Modern cryptography focuses on the idea of provable security [Ste03]. Provable security requires to define a formal communication model that describes how parties within the protocol, and an adversary. The capabilities of the adversary are formalized via the notion of oracle queries. No assumption is made on the specific attack method an adversary may use [KM07].

Security properties are defined as one or more games, each intended to capture a security property, played by the adversary within the pre-defined communication model. The protocol is proven secure by showing that a successful attack would imply a solution to an underlying intractable assumption, like the discrete log or the shortest vector problem. As long as the underlying problem is sufficiently hard, the adversary's advantage is negligible, and the protocol is guaranteed to resist attacks by any adversary who works within the communication model regardless of what specific attacks are mounted.

This allows us to pose trust on a handful of well-established hard problems, and have a mathematically rigorous theorem that establishes some guarantee of security (defined in a suitable way) under these problems.

Most of our protocols will be studied in the Common Reference String (CRS) model. The CRS model captures the assumption that a trusted setup in which all involved parties get access to the same output taken from some PPT algorithm G. Schemes proven secure in the CRS model are secure provided that the setup was performed correctly. If the output distribution of G is uniform, the CRS model is also seen as an acronym for Common Random String, to underline the fact that no particular structure on the CRS is required. The limits of the CRS model, particularly in the context of NIZKs, have been studied by Bellare, Fuchsbauer and Scafuro [BFS16], that studied what can be achieved when the CRS is set up maliciously. More recently, Groth et al. [GKM+18] have provided an alternative model where a number of users can update a universal CRS. The updatable CRS model guarantees security if at least one of the users updating the CRS is honest.

Real-world applications. Results in the scope of proof systems were considered mostly theoretical proofs of concept until recently, when several theoretical and practical breakthroughs [PHGR13, BBB+18] proved they could be effectively used (with modern hardware) for securing complex cryptographic protocols. As more versatile forms of zero knowledge become ever more practical, previously unrealizable applications are beginning to emerge: zero-knowledge proofs provide a means to send credit card details safely and securely to a trusted retail brand [Cha08]; electronic voting systems such as Helios [Adi08] use them to guarantee privacy while withholding security; electronic cash systems such as zCash [BCG+14], Monero, QuisQuis [FMMO18] use arguments of knowledge to guarantee some degree of privacy in money transactions; privacy-preserving services [DGS+18] are using them to guarantee anonymity of their users.

Additionally, in the past few years, a remarkable effort of standardization that attempts to unite members of industry and academia has also led to a series of documents for standardization proposals. For more information, we direct the reader towards https://zkproof.org.

1.3 Our results

We revisit non-interactive arguments of knowledge under two different aspects: first we explore what can be achieved in terms of privacy without any setup assumption nor random oralces; secondly, we explore what can be achieved using the current frameworks for constructing zero-knowledge proofs and post-quantum assumptions. To conclude, we illustrate how arguments of knowledge can be used to build privacy enhancing applications, and secure our digital communications.

Non-Interactive zaps of knowledge [FO18]. This work defines a witness-indistinguishable argument of knowledge in the standard model, that is, a non-interactive zap of knowledge. While non-interactive zero-knowledge (NIZK) proofs require trusted parameters, Groth, Ostrovsky and Sahai [GOS06a] constructed non-interactive witness-indistinguishable (NIWI) proofs without any setup; they called their scheme a non-interactive zap.

In Chapter 3, we provide the first NIWI argument of *knowledge without* parameters. Consequently, our scheme is also the first subversion-resistant knowledge-sound proof system, a notion recently proposed by Fuchsbauer [Fuc18]. This work extends the analysis of Bellare et al. [BFS16] on proof systems in the face of parameter subversion. We also present an alternative and more efficient ZAK based on Groth-Sahai proofs, which could be relevant for practical applications.

Designated-verifier, lattice-based zk-SNARKs [GMNO18]. To this day, zk-SNARKs are being used for delegating computation, electronic cryptocurrencies, and anonymous credentials. However, all current SNARKs implementations rely on assumptions that succumb to attacks on quantum computers (*pre-quantum*) and, for this reason, are not expected to withstand cryptanalytic efforts over the next few decades.

In Chapter 4, we introduce the first designated-verifier zk-SNARK based on lattice assumptions, which are believed to be post-quantum secure. We provide a generalization in the spirit of Gennaro et al. [GGPR13] to the SNARK of Danezis et al. [DFGK14] that is based on Square Span Programs (SSPs). We focus on designated-verifier proofs and propose a protocol in which a proof consists of just 5 LWE encodings. We provide a concrete choice of parameters as well as extensive benchmarks on a C implementation, showing that our construction is practically instantiable.

Aggregate Cash Systems and Mimblewimble [FOS19]. Mimblewimble is an electronic cash system proposed by an anonymous author in 2016 [Jed16]. It combines several privacy-enhancing techniques initially envisioned for Bitcoin, such as Confidential Transactions [Max15], non-interactive merging of transactions [SMD14a], and cut-through of transaction inputs and outputs [Max13b]. As a remarkable consequence, coins can be deleted once they have been spent while maintaining public verifiability of the ledger, which is not possible in Bitcoin. This results in tremendous space savings for the ledger and efficiency gains for new users, who must verify their view of the system.

In Chapter 5, we adopt a provable-security approach to analyze the security of Mimblewimble. We give a precise syntax and formal security definitions for an abstraction of Mimblewimble that we call an *aggregate cash system*. We then formally prove the security of Mimblewimble in this definitional framework. Our results imply in particular that two natural instantiations (with Pedersen commitments and Schnorr [Sch90] or BLS [BLS04] signatures) are provably secure against inflation and coin theft under standard assumptions.

1.4 Associated publications and other contributions

Besides what is exposed in this thesis, during my time as a PhD student I explored other cryptographic primitives, such as oblivious transfer, function secret sharing, hash functions that mapped to elliptic curve points, and anonymous tokens.

Conference Papers

Homomorphic Secret Sharing [BCG$^+$17]. A (2-party) Homomorphic Secret Sharing scheme splits an input x into shares (x_0, x_1) such that each share computationally hides x, and there exists an efficient homomorphic evaluation algorithm Eval such that for any function (or "program") P from a given class it holds that $\mathsf{Eval}(x_0, P) + \mathsf{Eval}(x_1, P) = P(x)$. HSS schemes were introduced by Boyle et al. [BGI16, BGI17]. In this work, we propose a number of theoretical and practical optimizations. We implemented both the naïve algorithms as well as the optimized version developed in this work, and provided exteded benchmarks as well as performance graphs.

Actively Secure 1-out-of-N OT Extension [OOS17]. Oblivious Transfer (OT) is a two-party protocol between a sender S and a receiver R. The sender transmits some of its inputs to the receiver, in such a way that its choice remains oblivious, and at the same time R cannot obtain more information than it is entitled to. For example, in its simplest flavor, 1-out-of-2 OT, a sender has two input messages $m_0 \in \{0,1\}^\lambda$ and $m_1 \in \{0,1\}^\lambda$ and learns nothing. The receiver inputs a choice bit b and learns only m_b. Oblivious Transfer extensions [Bea96, IKNP03] are protocols

that extend OT starting with a small number (say, security parameter λ) of "base" OTs, to create $\mathsf{poly}(\lambda)$ additional OTs using only symmetric primitives, with computational security λ.

In this work, we presented a novel 1-out-of-N oblivious transfer (OT) extension protocol with active security, which achieves very low overhead compared to the passively secure protocol of Kolesnikov and Kumaresan [KK13]. Our protocol obtains active security using a consistency check which requires only simple computation and has a communication overhead that is independent of the total number of OTs to be produced. We prove its security in both the random oracle model and the standard model, assuming a variant of correlation robustness. We describe an implementation, which demonstrates our protocol only incurs an overhead of around 5–30% on top of the passively secure protocol.

Furthermore, random 1-out-of-N OT is a key building block in recent, very efficient, passively secure private set intersection (PSI) protocols. Private set intersection allows two parties to compute the intersection of their sets without revealing anything except the intersection. Our random OT extension protocol has the interesting feature that it even works when N is exponentially large in the security parameter, provided that the sender only needs to obtain polynomially many outputs. We show that this can be directly applied to improve the performance of PSI, allowing the core private equality test and private set inclusion subprotocols to be carried out using just a single OT each. This leads to a reduction in communication of up to 3 times for the main component of PSI.

Internet Standards

In addition to the academic publications, we helped with (minor) contributions to the following IETF drafts:

Hashing to the curve [FHSS+19]. Many cryptographic protocols require a procedure to hash an arbitrary input to a point on an elliptic curve, e.g. Password Authenticated Key Exchange [BMP00], Identity-Based Encryption [BF01] and Boneh-Lynn-Shacham signatures [BLS01]. This procedure is known as hashing to the curve; i.e., a hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{G}$.

Unfortunately for implementors, the precise hash function that is suitable for a given scheme is not necessarily included in the description of the protocol. Compounding this problem is the need to pick a suitable curve for the specific protocol. Together with Brice Minaud and an undergraduate student Anita Dürr, we contributed to the standard and added a section about fast hashing into BLS curves [BLS03] and fast cofactor multiplication [SBC+09, FKR12, BP17].

Anonymous Tokens [DS19]. Anonymous tokens are lightweight authorization mechanisms that can be useful in quickly assessing the reputation of a client in latency-sensitive communication.

Increasingly the stability of our information-based world depends on trusted, reliable sources of content. An unwanted by-product of the growth of Content Delivery Networks (CNDs) is that a handful of organizations are becoming global arbiters for which content requests are allowed and which are blocked in an attempt to stanch malicious traffic. In particular, users hiding behind privacy-enhancing tools such as Tor can be unfairly targeted by anti-spam or anti-DoS countermeasures. Anonymous Tokens we provide a solution to prevent users from being exposed to a disproportionate amount of internet challenges such as CAPTCHAs [DGS+18], and more generally to check that a client has been previously authorized by a service without learning any other information.

Once standardized, such lightweight authorization mechanisms will be useful in quickly assessing the reputation of a client in latency-sensitive communication. As a part of an internship at Google under the supervision of Mariana Raykova and Tancrède Lepoint, I had the chance to assess and extend the scheme in the process of standardization.

Chapter 2

Preliminaries

In this chapter, we introduce the notation and basic assumptions and primitives employed throughout this manuscript. We start by recalling some standard mathematical and computational notions, then we briefly introduce provable security. We also recall some well-known number-theoretic assumptions, to introduce the cryptographic primitives used throughout this work.

2.1 Notation

Sets, integers, moduli, and associated rings and groups. We denote real numbers by $\mathbb{R}$, integers by $\mathbb{Z}$ and non-negative integers by $\mathbb{N}$. If a, b are two integers such that $a < b$, we denote the (closed) integer interval from a to b by $[\![a, b]\!]$. We use $[b]$ as shorthand for $[\![1, b]\!]$.

If q is a positive integer, we denote by $\mathbb{Z}_q$ the ring of integers modulo q. Often, q will be an odd prime; in these cases, we will consider the ring $\mathbb{Z}_q$ just as an additive group $(\mathbb{Z}_q, +)$ or just consider the multiplicative subgroup $(\mathbb{Z}_q^*, \cdot)$ of order $\varphi(q) = q - 1$, where φ is Euler's totient function.

In all of our constructions, the order of $\mathbb{Z}_q$ will be public. Therefore, elements of $\mathbb{Z}_q$ are represented as integers of the set $[\![0, q-1]\!]$. For an integer $x \in \mathbb{Z}$, $x \mod q$ is the reminder of the Euclidean division of x by q. It can be seen both as an integer in $[\![0, q-1]\!]$ and as an element of $\mathbb{Z}_q$. Vectors are denoted by lower-case bold letters, like $\mathbf{v}$, and are always columns. We indicated a vector $\mathbf{v}$'s entry by v_i (not in bold). We use $\mathbf{v}^T$ to denote the transpose of a vector $\mathbf{v}$.

Random variables. For a random variable X over a probability space $(\Omega, \Sigma, \Pr)$, and a possible outcome x, we write $\Pr[X = x]$ to indicate the measure of the preimage of $\{x\} \in \Sigma$ under $\Pr$. We denote the action of sampling x from X with $x \leftarrow X$. Let us call *range* of a random variable $X : \Omega \to E$ the set of elements $x \in E$ for which the probability that X has outcome x is strictly positive. In this work, we refer only to random variables whose range is finite.

A random variable X defined on a finite probability space $(\Omega, \Pr)$ is said to have the *uniform distribution* if $\Pr[X = x] = 1/|\operatorname{Im}(X)|$ where $\operatorname{Im}(X)$ denotes the image of X. Given a non-empty finite set S, we let $x \leftarrow\!\!\$\, S$ denote the operation of sampling an element x from S uniformly at random.

Asymptotics. Given a function $f : \mathbb{N} \to \mathbb{R}$, the set $O(f)$ describes all functions that can be asymptotically upper-bounded by f, that is, all g such that $\exists c, \lambda_0 \in \mathbb{N},\ 0 \leq g(\lambda) \leq cf(\lambda)$ for all $\lambda \geq \lambda_0$. With a slight abuse of notation, we write $g = O(f)$ to denote that $g \in O(f)$. The set $\tilde{O}(f)$ describes all functions that can be upper-bounded by f ignoring logarithmic factors, that is, all g such that $\exists c, k, \lambda_0 \in \mathbb{N},\ 0 \leq g(\lambda) \leq cf(\lambda)\log^k(\lambda)$ for all $\lambda \geq \lambda_0$. We use the notation $o(f)$ to identify all functions that can be asymptotically upper-bounded by f, where the upper-bound is strict. That is, all g such that $\forall c\ \exists \lambda_0 \in \mathbb{N},\ 0 \leq g(\lambda) < cf(\lambda)$ for all $\lambda \geq \lambda_0$.

A function $\mu\colon \mathbb{N} \to [0,1]$ is negligible (denoted $\mu = \mathsf{negl}(\lambda)$) if $\mu(\lambda) = o(\lambda^{-c})$ for any fixed constant c. That is, if for all $c \in \mathbb{N}$ there exists $\lambda_c \in \mathbb{N}$ such that $\mu(\lambda) < \lambda^{-c}$ for all $\lambda \geq \lambda_c$. A function ν is *overwhelming* if $1 - \nu = \mathsf{negl}(\lambda)$. We let $\mathsf{poly}(\lambda)$ denote the set of polynomials in λ (more precisely, functions upper-bounded by a polynomial in λ) with integer coefficients.

Languages, machines, function families and complexity classes. Languages are denoted in calligraphic, e.g. $\mathcal{L}$. We focus on languages whose alphabet is the set of bits $\{0,1\}$.

Algorithms are formalized as Turing Machines and denoted in serif, e.g. M. We let $\mathsf{M.rl}(\lambda)$ be a *length function* (i.e., a function $\mathbb{N} \to \mathbb{N}$ polynomially bounded) in λ defining the length of the randomness for a probabilistic interactive Turing Machine M. By $y \coloneqq \mathsf{M}(x_1, \ldots; r)$ we denote the operation of running algorithm M on inputs $x_1, \ldots$ and coins $r \in \{0,1\}^{\mathsf{M.rl}(\lambda)}$ and letting y denote the output. We see M both as a Turing Machine as well as a random variable. By $y \leftarrow \mathsf{M}(x_1, \ldots)$, we denote $y \coloneqq \mathsf{M}(x_1, \ldots; r)$ for random $r \in \mathsf{M.rl}(\lambda)$, and $[\mathsf{M}(x_1, \ldots)]$ the range of M on inputs $x_1, \ldots$. Unless otherwise specified, all the algorithms defined throughout this work are assumed to be probabilistic Turing machines that run in time $\mathsf{poly}(\lambda)$ – i.e., in PPT. An adversary is denoted by $\mathcal{A}$; when it is interacting with an oracle $\mathcal{O}$, we write $\mathcal{A}^{\mathcal{O}}$. For two PPT machines A, B, with $(\mathsf{A} \| \mathsf{B})(x)$ we denote the execution of A followed by the execution of B on the same input x and with the same random coins. The output of the two machines is concatenated and separated with a semicolon, e.g., $(out_{\mathsf{A}}; out_{\mathsf{B}}) \leftarrow (\mathsf{A} \| \mathsf{B})\,(x)$.

We say that a language $\mathcal{L}$ is in the complexity class P if there exists a deterministic Turing machine M that can recognize $\mathcal{L}$ in polynomial time. We say that $\mathcal{L}$ is in NP if there exists a non-deterministic Turing machine M that can recognize $\mathcal{L}$ in polynomial time. For a language $\mathcal{L}$, we denote with $\mathsf{R}_{\mathcal{L}} \subseteq \mathcal{L} \times \{0,1\}^*$ the relation associated to $\mathcal{L}$. For a pair $(\phi, w) \in \mathsf{R}_{\mathcal{L}}$, we call $\phi \in \mathcal{L}$ the *statement* and $w \in \{0,1\}^*$ the *witness*. A language $\mathcal{L}$ is NP-complete if it there is a reduction to it that runs in polynomial time.

In this manuscript, we will mostly work with the language of circuit satisfiability, which is NP-complete. For a binary circuit C, the set $\mathsf{R}(\mathsf{C})$ is the set of inputs w that satisfy $\mathsf{C}(w) = 1$. Without loss of generality, we assume that circuits consist solely of NAND gates. Unless otherwise specified, all following algorithms are assumed to be randomized and to run in time $\mathsf{poly}(\lambda)$. Unless otherwise stated, we only consider uniform machines to model the adversary $\mathcal{A}$ and the extractor Ext, in the spirit of Goldreich et al. [Gol93, BFS16]. We use the names "machine", "Turing Machine", and "algorithm" interchangeably. A machine $\mathcal{A}$ that outputs a boolean 1 or 0 is called a *distinguisher*. If an algorithm calls a subroutine which returns $\perp$, we assume it stops and returns $\perp$ (this does not hold for an adversary calling an *oracle* which returns $\perp$).

2.2 Provable security

Security notions of cryptographic assumptions are often described as *expermiments* or *games* where an adversary is called with one or several inputs, and potentially given oracle access to some subprocedures. A security experiment is seen both as an algorithm and as a random variable (more exactly, as a process indexed in $\lambda \in \mathbb{N}$).

Some of our security notions consist in distinguishing two experiments $\mathrm{EXP}^0(\lambda)$ and $\mathrm{EXP}^1(\lambda)$. In this case, we need a measure of the ability of $\mathcal{A}$ to distinguish $\mathrm{EXP}^0_{\mathcal{A}}(\lambda)$ and $\mathrm{EXP}^1_{\mathcal{A}}(\lambda)$.

Definition 2.1 (Statistical Distance)**.** *Let $X_0(\lambda), X_1(\lambda)$ be two families of random variables indexed by $\lambda \in \mathbb{N}$, with range R. The statistical distance between X_0 and X_1 is a map $\mathbb{N} \to [0,1]$:*

$$\Delta(X_0(\lambda), X_1(\lambda)) \coloneqq \frac{1}{2} \sum_{x \in R} \left| \Pr[X_0(\lambda) = x] - \Pr[X_1(\lambda) = x] \right|.$$

It can be easily shown that the statistical distance is a metric on probability distributions. We say that X_0 and X_1 are *perfectly indistinguishable* if $\Delta(X_0, X_1) = 0$. We say that X_1 and X_1 are *statistically indistinguishable* if $\Delta(X_0, X_1)$ is negligible in λ. It can be shown that perfect and statistical indistinguishability are an equivalence relation for families of random variables indexed in $\lambda \in \mathbb{N}$ with the same range R.

Definition 2.2. *Let $\mathcal{A}$ be a Turing machine, and $\mathrm{EXP}^b_{\mathcal{A}}(\lambda)$ (with $b \in \{0,1\}$) be two families of random variables indexed in $\lambda \in \mathbb{N}$ with range R. The* advantage of $\mathcal{A}$ in distinguishing *the experiments $\mathrm{EXP}^b_{\mathcal{A}}(\lambda)$ (for $b \in \{0,1\}$) is a map $\mathbb{N} \to [0,1]$ defined as:*

$$\begin{aligned}\mathsf{Adv}^{\mathrm{exp}}_{\mathcal{A}}(\lambda) &\coloneqq \Delta(\mathrm{EXP}^0_{\mathcal{A}}(\lambda),\ \mathrm{EXP}^1_{\mathcal{A}}(\lambda)) \\ &= \left|\Pr\left[\mathrm{EXP}^0_{\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathrm{EXP}^1_{\mathcal{A}}(\lambda) = 1\right]\right|\end{aligned}$$

The last equality can be shown applying the definition of statistical distance and using the negation rule. If two random variables are perfectly indistinguishable, then the advantage of any unbounded distinguisher $\mathcal{A}$ is zero. If two random variables are statistically indistinguishable, then the advantage of any unbounded distinguisher $\mathcal{A}$ is negligible. We say that two random variables are *computationally indistinguishable* if the advantage of any PPT algorithm in distinguishing $\mathrm{EXP}^0_{\mathcal{A}}(\lambda)$ and $\mathrm{EXP}^1_{\mathcal{A}}(\lambda)$ is negligible. It can be shown that computational indistinguishability is an equivalence relation for families of random variables indexed in $\lambda \in \mathbb{N}$ with the same range R.

Sometimes, we are interested in studying the success probability of an adversary in winning a computational security experiment. In these cases, as opposed to indistinguishability-based notions, we study the probability that at the end of the interaction the adversary outputs the solution for a problem believed to be computationally hard.

Definition 2.3. *Let $\mathrm{EXP}_{\mathcal{A}}(\lambda)$ be a family of random variables indexed in $\lambda \in \mathbb{N}$ and parameterized in a Turing machine $\mathcal{A}$, with range $\{0,1\}$. The* advantage of $\mathcal{A}$ in winning $\mathrm{EXP}_{\mathcal{A}}(\lambda)$ *is a map $\mathbb{N} \to [0,1]$ defined as:*

$$\mathsf{Adv}^{\mathrm{exp}}_{\mathcal{A}}(\lambda) \coloneqq \Pr[\mathrm{EXP}_{\mathcal{A}}(\lambda) = 1]$$

Sometimes (for instance, in Definition 5.5, where a lot of security experiments are displayed), we will denote the security experiment within the probability measure, e.g.:

$$\mathsf{Adv}^{\mathrm{exp}}_{\mathcal{A}}(\lambda) \coloneqq \Pr[\textit{code-of-experiment} : \textit{winning-condition}]\,.$$

In the rest of this manuscript, we will say that a notion holds perfectly if the advantage associated to the security experiment is zero, for any unbounded adversary $\mathcal{A}$. We will say that a notion holds statistically if the advantage associated to the security experiment is negligible in λ, for any unbounded adversary $\mathcal{A}$. We will say that a notion holds computationally if the advantage associated to the security experiment is negligible in λ, for any PPT adversary $\mathcal{A}$.

Proofs by hybrid arguments

Most of our security proofs are proofs by hybrid arguments as defined in [BR06, Sho01] to bound the success probability of an adversary $\mathcal{A}$ in some game $\mathrm{EXP}_{\mathcal{A}}(\lambda)$ corresponding to a certain security notion.

Roughly speaking, in instistinguishability-based notions, we bound the advantage of the adversary by constructing a sequence of experiments where $\mathrm{EXP}^0_{\mathcal{A}}(\lambda)$ will be the process associated to the first experiment and $\mathrm{EXP}^1_{\mathcal{A}}(\lambda)$ to the last one. We will prove that two consecutive games are perfectly, statistically, or computationally indistinguishable. In other words, we bound the difference of success probabilities between two consecutive games by a negligible quantity. By the

triangular inequality, it will follow that the security experiments $\mathrm{EXP}^0_{\mathcal{A}}(1^\lambda)$ and $\mathrm{EXP}^1_{\mathcal{A}}(1^\lambda)$ are indistinguishable, except with negligible probability, for the adversary $\mathcal{A}$.

Similarly, in order to bound the probability of an adversary $\mathcal{A}$ in winning an experiment $\mathrm{EXP}_{\mathcal{A}}(\lambda)$, we construct a sequence of hybrids where the first game is $\mathrm{EXP}_{\mathcal{A}}(\lambda)$, and the last game corresponds to some security assumption or is such that the adversary just cannot win.

2.3 Cryptographic primitives

We denote by $\lambda \in \mathbb{N}$ the security parameter. In Chapter 4, when exhamining concrete security, we will specify another parameter to indicate statistical security, $\kappa \in \mathbb{N}$. Throughout this work, we assume the existence of a PPT algorithm Pgen that, given as input the security parameter in unary 1^λ, outputs a set of parameters Γ. This can be, concretely, the description of a finite cyclic group of prime order where computing discrete logarithm is hard, a symmetric or asymmetric bilinear group description, or a set of lattice parameters. For the sake of simplicity, we will denote all our schemes w.r.t. a set of parameters Γ and assume that the security parameter ($\lambda \in \mathbb{N}$ such that $\Gamma \leftarrow \mathsf{Pgen}(1^\lambda)$) can be derived from Γ.

Commitment schemes

A commitment scheme Com consists of the following three algorithms:

- $\sigma \leftarrow \mathsf{Com.G}(\Gamma)$, the CRS generation algorithm, outputs a CRS σ.
- $(C, r) \leftarrow \mathsf{Com.C}(\sigma, v)$, the commitment algorithm, outputs a commitment C to the given value v together with the *opening information* r.
- $bool \leftarrow \mathsf{Com.O}(\sigma, C, v, r)$, the opening algorithm, outputs 1 if C is a commitment to v witnessed by r, and 0 otherwise.

We denote with $\mathcal{V}_{\mathsf{cp}}$ the space set of values, with $\mathcal{R}_{\mathsf{cp}}$ the set of possible openings, and with $\mathcal{C}_{\mathsf{cp}}$ the set of commitments. In this work, $\mathsf{Com.C}$ will always return the randomness used, and $\mathsf{Com.O}$ simply recomputes the commitment and checks that $C = \mathsf{Com.C}(\sigma, V; r)$. Consequently, *correctness* of the scheme is trivial. To ease notation for commitments and openings, we will always assume that the set of parameters Γ can be deduced from σ, and omit the opening information from the returned value if specified within the random coins.

Generally, we require commitment schemes to be *hiding* and *binding*. Loosely speaking, a scheme is *hiding* if the commitment C reveals no information about v. A scheme is *binding* if a cheating committer cannot change its mind about the value it committed to. Formally, it is hard to find C, v, r, v' and r' such that $v \neq v'$ and $\mathsf{Com.O}(\sigma, C, v, r) = 1 = \mathsf{Com.O}(\sigma, C, v', r')$.

Definition 2.4 (Hiding)**.** *A commitment scheme* Com *is* hiding *if for any* PPT *adversary* $\mathcal{A}$*:*

$$\mathsf{Adv}^{\mathrm{hid}}_{\mathsf{Com},\mathcal{A}}(\lambda) \coloneqq \left|\Pr\left[\mathrm{HID}^0_{\mathsf{Com},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathrm{HID}^1_{\mathsf{Com},\mathcal{A}}(\lambda) = 1\right]\right| = \mathsf{negl}(\lambda)\,,$$

where the game $\mathrm{HID}_{\mathsf{Com},\mathcal{A}}(\lambda)$ *is defined in Fig. 2.1.*

Definition 2.5 (Binding)**.** *A commitment scheme* Com *is* binding *if for any* PPT *adversary* $\mathcal{A}$*:*

$$\mathsf{Adv}^{\mathrm{bnd}}_{\mathsf{Com},\mathcal{A}}(\lambda) \coloneqq \Pr\left[\mathrm{BND}_{\mathsf{Com},\mathcal{A}}(\lambda) = 1\right] = \mathsf{negl}(\lambda)\,,$$

where the game $\mathrm{BND}_{\mathsf{Com},\mathcal{A}}(\lambda)$ *is defined in Fig. 2.1.*

Game $\mathrm{HID}^b_{\mathsf{Com},\mathcal{A}}(\lambda)$

$\Gamma \leftarrow \mathsf{Pgen}(1^\lambda)$
$\sigma \leftarrow \mathsf{Com.G}(\Gamma)$
$b' \leftarrow \mathcal{A}^{\textsc{Commit}}(\sigma)$
return b'

Oracle $\textsc{Commit}(v_0, v_1)$

$(C, r) \leftarrow \mathsf{Com.C}(\sigma, v_b)$
return C

Game $\mathrm{BND}_{\mathsf{Com},\mathcal{A}}(\lambda)$

$\Gamma \leftarrow \mathsf{Pgen}(1^\lambda)$
$\sigma \leftarrow \mathsf{Com.G}(\Gamma)$
$(C, v_0, r_0, v_1, r_1) \leftarrow \mathcal{A}(\sigma)$
return $v_0 \neq v_1$ **and**
$\mathsf{Com.O}(\sigma, C, v_0, r_0)$ **and** $\mathsf{Com.O}(\sigma, C, v_1, r_1)$

Figure 2.1: The games for hiding (HID) and binding (BND) of a commitment scheme Com.

Throughout this work, and more specifically in Chapter 3, we also require a perfectly binding commitment scheme to be *extractable*, that is, Com is equipped with a setup algorithm $\mathsf{Com.G}$ that in addition to σ returns also some trapdoor τ, and there exists an efficient extraction algorithm $\mathsf{Com.E}$ that, given as input the trapdoor information τ, recovers the value v to which C is bound.

Proof systems

A (non-interactive) proof system Π for a relation R consists of the following three algorithms:

- $(\sigma, \mathsf{vrs}, \tau) \leftarrow \Pi.\mathsf{G}(\Gamma)$ the CRS generation algorithm takes as input some set of parameters Γ and outputs a tuple consisting of: a common reference string σ that will be given publicly; a designated verifier string vrs, that will be used for verifying; and a trapdoor τ that will be useful for proving security of the proof system.

- $\pi \leftarrow \Pi.\mathsf{P}(\sigma, \phi, w)$, a prover which takes as input some $(\phi, w) \in \mathsf{R}$ and a CRS σ, and outputs a proof π.

- $bool \leftarrow \Pi.\mathsf{V}(\mathsf{vrs}, \phi, \pi)$ a verifier that, given as input a statement ϕ together with a proof π outputs 1 or 0, indicating acceptance of the proof.

In most instances, we will be dealing with *publicly verifiable protocols*. In publicly verifiable proofs, the verifier key vrs is equal to the CRS σ; in those cases, we will omit it from the output of the setup algorithm. In this manuscript, we will often deal with *families* of relations, i.e. relations R_δ parametrized by some $\delta \in \mathbb{N}$. For those, we assume that the proof system Π is defined over the family of relations $\mathsf{R} = \{\mathsf{R}_\delta\}_\delta$ and that the setup algorithm $\Pi.\mathsf{G}$ takes an additional parameter δ which specifies the particular relation used during the protocol (and which is included in the returned CRS). For instance, in Chapter 5, when describing proofs for a certain range $[\![0, v_{\max}]\!]$, the proof system will be defined over a relation $\mathsf{R}_{v_{\max}}$, where $v_{\max}$ is the maximum integer allowed.

Generally, we require proof systems to be *complete* and *sound*. A proof system is complete if every honestly-generated proof verifies.

Definition 2.6 (Completeness)**.** *A non-interactive proof system Π for relation R is (computationally)* complete *if for any PPT adversary $\mathcal{A}$:*

$$\mathsf{Adv}^{\mathrm{compl}}_{\Pi,\mathsf{R},\mathcal{A}}(\lambda) := \Pr\left[\mathrm{COMPL}_{\Pi,\mathsf{R},\mathcal{A}}(\lambda) = 1\right] = \mathsf{negl}(\lambda)\,,$$

where $\mathrm{COMPL}_{\Pi,\mathsf{R},\mathcal{A}}(\lambda)$ *is the game depicted in Fig. 2.2.*

Game $\mathrm{COMPL}_{\Pi,\mathsf{R},\mathcal{A}}(\lambda)$

$\Gamma \leftarrow \mathsf{Pgen}(1^\lambda)$
$(\sigma, \mathsf{vrs}, \tau) \leftarrow \Pi.\mathsf{G}(\Gamma)$
$(\phi, w) \leftarrow \mathcal{A}(\sigma)$
$\pi \leftarrow \Pi.\mathsf{P}(\sigma, \phi, w)$
return $(\Pi.\mathsf{V}(\mathsf{vrs}, \phi, \pi) = 0$ **and** $(\phi, w) \in \mathsf{R})$

Game $\mathrm{KSND}_{\Pi,\mathsf{R},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)$

$\Gamma \leftarrow \mathsf{Pgen}(1^\lambda)$
$(\sigma, \mathsf{vrs}, \tau) \leftarrow \Pi.\mathsf{G}(\Gamma)$
$(\phi, \pi; w) \leftarrow (\mathcal{A} \| \mathsf{Ext})^{\Pi.\mathsf{V}(\mathsf{vrs}, \sigma, \cdot)}(\sigma)$
return $(\Pi.\mathsf{V}(\mathsf{vrs}, \phi, \pi) = 1$ **and** $(\phi, w) \notin \mathsf{R})$

Game $\text{S-EXT}_{\Pi,\mathsf{R},\mathcal{A}}(\lambda)$

$\mathsf{Q} \coloneqq (\,)\,;\ \Gamma \leftarrow \mathsf{Pgen}(1^\lambda)$
$(\sigma, \tau) \leftarrow \Pi.\mathsf{G}(\Gamma)$
$(\phi, \pi) \leftarrow \mathcal{A}^{\textsc{Prove}}(\sigma)$
for $i = 1 \ldots |\mathbf{u}|$ **do**
  $w_i \coloneqq \Pi.\mathsf{Ext}(\sigma, \tau, \phi_i, \pi_i)$
return $\bigvee_{i=1}^{|\mathbf{u}|} \big(\Pi.\mathsf{V}(\mathsf{vrs}, \phi_i, \pi_i)$ **and** $(\phi_i, \pi_i) \notin \mathsf{Q}$ **and** $(\phi_i, w_i) \notin \mathsf{R}\big)$

Oracle $\textsc{Prove}(u)$

$\pi \leftarrow \Pi.\mathsf{Sim}(\sigma, \tau, \phi)$
$\mathsf{Q} \coloneqq \mathsf{Q} \,\|\, ((\phi, \pi))$
return π

Figure 2.2: Games for completeness (COMPL), knowledge soundness (KSND), and simulation-extractability soundness (S-EXT).

Soundness. The central security property of a proof system is *soundness*, that is, no adversary can produce a proof for a false statement. A stronger notion is *knowledge-soundness*. Knowledge soundness [BG93] means that for any prover able to produce a valid proof, there exists an efficient algorithm which, when given the same inputs as the prover (and the same random coins), is capable of extracting a witness for the given statement. This algorithm is called *knowledge extractor*, and it embodies the concept that the prover "must know" a witness for the given statement. Formally:

Definition 2.7 (Knowledge Soundness)**.** *A non-interactive proof system* Π *for relation* R *is* knowledge-sound *if for any* PPT *adversary* $\mathcal{A}$ *there exists a* PPT *extractor* Ext *such that:*

$$\mathsf{Adv}^{\mathrm{ksnd}}_{\Pi,\mathsf{R},\mathcal{A},\mathsf{Ext}}(\lambda) \coloneqq \Pr\Big[\mathrm{KSND}_{\Pi,\mathsf{R},\mathcal{A},\mathsf{Ext}}(\lambda) = 1\Big] = \mathsf{negl}(\lambda)\,,$$

where $\mathrm{KSND}_{\Pi,\mathsf{R},\mathcal{A},\mathsf{Ext}}(\lambda)$ *is defined in Figure 2.2.*

An *argument of knowledge* is a *knowledge-sound* proof system [BLCL91]. If the adversary is computationally unbounded, we speak of *proofs* rather than arguments.

Remark 2.8. Note that the verification oracle is relevant only for designated-verifier proof systems, as in publicly verifiable protocols the adversary is provided as input the CRS and can simply run the verification algorithm. An important consideration that arises when defining knowledge soundness in the designated-verifier setting is whether the adversary should be granted access to a verification oracle. Pragmatically, allowing the adversary to query a verification oracle captures the fact that VRS can be reused $\mathsf{poly}(\lambda)$ times. While this distinction cannot be made in the publicly verifiable setting, the same is not true for the designated-verifier setting. In the specific case of our construction, we formulate and prove our protocol allowing the adversary access to the verification algorithm (which has been named *strong soundness* in the past [BISW17]), and later discuss which optimizations can be made when using the weaker notion of soundness, where the adversary cannot access the verification oracle.

Game $\mathrm{ZK}^b_{\Pi,\mathsf{R},\mathcal{A}}(\lambda)$

$\Gamma \leftarrow \mathsf{Pgen}(1^\lambda)$
$(\sigma,\tau) \leftarrow \Pi.\mathsf{G}(\Gamma)$
$b' \leftarrow \mathcal{A}^{\textsc{Prove}}(\sigma)$
return b'

Oracle $\textsc{Prove}(\phi, w)$

if $(\phi, w) \notin \mathsf{R}$ **then return** $\perp$
$\pi_0 \leftarrow \Pi.\mathsf{P}(\sigma, \phi, w)$
$\pi_1 \leftarrow \Pi.\mathsf{Sim}(\sigma, \tau, \phi)$
return π_b

Game $\mathrm{WI}^b_{\Pi,\mathsf{R},\mathcal{A}}(\lambda)$

$\Gamma \leftarrow \mathsf{Pgen}(1^\lambda)$
$(\sigma,\tau) \leftarrow \Pi.\mathsf{G}(\Gamma)$
$b' \leftarrow \mathcal{A}^{\textsc{Prove}}(\sigma)$
return b'

Oracle $\textsc{Prove}(\phi, w_0, w_1)$

if $(\phi, w_0) \notin \mathsf{R}$ **or** $(\phi, w_1) \notin \mathsf{R}$:
  return $\perp$
$\pi \leftarrow \Pi.\mathsf{P}(\sigma, \phi, w_b)$
return π

Figure 2.3: Games for zero knowledge (ZK), and witness indistinguishability (WI).

In security proofs where the reduction simulates certain proofs, knowledge-soundness might not be sufficient. The stronger notion *simulation-extractability* guarantees that even then, from every proof output by the adversary, $\Pi.\mathsf{Ext}$ can extract a witness. Note that we define a multi-statement variant of simulation extractability: the adversary returns a list of statements and proofs and wins if there is at least one statement such that the corresponding proof is valid and the extractor fails to extract a witness.

Definition 2.9 (Simulation-Extractability)**.** *A non-interactive proof system* Π *for relation* R *is* (multi-statement) simulation-extractable *if there exists an extractor* $\Pi.\mathsf{Ext}$ *such that, for any* PPT *adversary* $\mathcal{A}$*,*

$$\mathsf{Adv}^{\text{s-ext}}_{\Pi,\mathsf{R},\mathcal{A}}(\lambda) \coloneqq \Pr\left[\text{S-EXT}_{\Pi,\mathsf{R},\mathcal{A}}(\lambda) = 1\right] = \mathsf{negl}(\lambda)\,,$$

*where*S-EXT *be as defined in Fig. 2.2.*

Zero knowledge. A proof system Π is *zero-knowledge* if proofs leak no information about the witness. More precisely, Π specifies an additional PPT algorithm $\Pi.\mathsf{Sim}$ that takes as input the trapdoor information τ and a statement u, and outputs a valid proof π indistinguishable from those generated via $\Pi.\mathsf{P}$.

We define a *simulator* $\Pi.\mathsf{Sim}$ for a proof system Π as:

- $\pi^* \leftarrow \Pi.\mathsf{Sim}(\sigma, \tau, \phi)$: the simulated prover algorithm takes as input a CRS, a trapdoor τ, and a statement ϕ and outputs a simulated proof π^*.

Definition 2.10 (Zero Knowledge)**.** *A non-interactive proof system* Π *for relation* R *is* zero-knowledge *if there exists a simulator* $\Pi.\mathsf{Sim}$ *such that, for any* PPT *adversary* $\mathcal{A}$*:*

$$\mathsf{Adv}^{\text{zk}}_{\Pi,\mathsf{R},\mathcal{A}}(\lambda) = \left|\Pr\left[\mathrm{ZK}^0_{\Pi,\mathsf{R},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathrm{ZK}^1_{\Pi,\mathsf{R},\mathcal{A}}(\lambda) = 1\right]\right| = \mathsf{negl}(\lambda)\,,$$

where $\mathrm{ZK}^b_{\Pi,\mathsf{R},\mathcal{A}}(\lambda)$ *is defined in Figure 2.3.*

Feige and Shamir [FS90] proposed a relaxation of zero knowledge called *witness indistinguishability*, which only requires that it is indistinguishable which witness was used to compute a proof.

In other words, witness indistinguishability states that no PPT adversary can tell which of any two possible witnesses has been used to construct a proof.

Definition 2.11. *A proof system* Π *for relation* R *is witness-indistinguishable (WI) if, for any* PPT *adversary* $\mathcal{A}$*:*

$$\mathsf{Adv}^{\mathrm{wi}}_{\Pi,\mathsf{R},\mathcal{A}}(\lambda) \coloneqq \left|\Pr\left[\mathrm{WI}^{0}_{\Pi,\mathsf{R},\mathcal{A}}(\lambda)=1\right]-\Pr\left[\mathrm{WI}^{1}_{\Pi,\mathsf{R},\mathcal{A}}(\lambda)=1\right]\right| = \mathsf{negl}(\lambda)\,,$$

and $\mathrm{WI}^{b}_{\Pi,\mathsf{R},\mathcal{A}}(\lambda)$ *is the game depicted in Fig. 2.3.*

Zaps. A (non-interactive) zap is a *witness-indistinguishable*, non-interactive proof system: the prover simply sends a single message to deliver the whole proof, without any setup needed. The proof system thus reduces to a pair (P,V) or can be considered as defined above, but with a CRS generation algorithm that always outputs $\perp$. In Chapter 3, we will introduce non-interactive zaps of knowledge, that is, non-interactive zaps that are *arguments of knowledge* (cf. Definition 2.7).

SNARKs. A non-interactive proof system Π for a relation R is *succinct* if the proof has size quasi-linear in the security parameter, i.e., $|\pi| = \tilde{O}(\lambda)$. A *succinct non-interactive argument of knowledge* (SNARK) is a non-interactive proof system that is complete, succinct, and knowledge-sound. A zk-SNARK is a SNARK that satisfies zero knowledge (cf. Definition 2.10).

Chapter 3

Arguments of knowledge without setup: ZAKs

This work was published in the proceedings of the 16th International Conference on Applied Cryptography and Network Security, ACNS. It was completed with co-author Georg Fuchsbauer, and awarded "best student paper" with a monetary compensation from Springer.

Motivated by the subversion of trusted public parameters in standardized cryptographic protocols led by mass-surveillance activities, Bellare, Fuchsbauer and Scafuro [BFS16] investigate what security properties can be maintained for NIZK when its trusted parameters are subverted. CRS's for NIZK are especially easy to subvert, since they must be subvertible by design: zero knowledge requires that an honest CRS must be indistinguishable from a backdoored one, where the backdoor is the trapdoor used to simulate proofs.

Bellare et al. defined multiple security properties that protect against parameter subversion: subversion soundness (S-SND) means that no adversary can generate a malicious CRS together with a valid proof for a false statement; subversion zero knowledge (S-ZK) requires that even if the adversary sets up the CRS, there exists a simulator able to produce its full view; and subversion witness indistinguishability (S-WI) formalizes that even for proofs that were made under a subverted CRS, it is still infeasible to tell which of two witnesses was used.

Following Goldreich and Oren [GO94], Bellare et al. [BFS16] also showed that it is impossible to achieve subversion soundness and (standard) zero knowledge simultaneously. For subversion-sound proof systems, subversion witness indistinguishability is thus the best one can hope for. The authors [BFS16] observe that since proof systems that do not rely on a CRS cannot succumb to CRS-subversion attacks, non-interactive zaps [GOS06a] achieve both S-SND and S-WI.

Bellare et al. did not consider the stronger notion of knowledge soundness, which is the notion achieved by SNARKs, and which in many applications is the required notion for the used proof systems. For example, for all kinds of anonymous authentication, users prove knowledge of signatures (often called *certificates* or *credentials*, depending on the context); in this case soundness is not sufficient, as signatures always exist, but in the security proof they must actually be extracted in order to rely on their unforgeability. Fuchsbauer [Fuc18] has recently defined a subversion-resistant notion of knowledge soundness but left it open to give a scheme that achieves it. Such a scheme would protect against possible parameter subversion in any context where proving knowledge of a witness is required. In this chapter:

(i) We provide the first non-interactive zap with knowledge soundness; that is, a witness-indistinguishable proof system without parameters for which there exists an extractor that

recovers a witness from every valid proof.

(ii) Our zap is also the first fully subversion-resistant WI argument-of-knowledge system. In particular, it satisfies the recently defined notion of subversion knowledge soundness [Fuc18], as well as subversion witness indistinguishability [BFS16] (the strongest notion compatible with S-SND).

Bellare et al. [BFS16] introduce a new type of knowledge-of-exponent assumption, which they call DH-KE. They prove (standard) soundness and subversion zero knowledge of their main construction under DH-KE and the decision linear assumption (DLin) [BBS04]. Our construction builds on the DLin-based non-interactive zap from [GOS06a], whose soundness we upgrade to knowledge soundness, assuming DH-KE. As for this zap, the language of our proof system is circuit satisfiability and thus universal. Groth, Ostrovsky and Sahai's [GOS06a] starting point is a "dual-mode" [GOS06b, PVW08] non-interactive proof system, for which there are two indistinguishable types of CRS: one leading to proofs that are perfectly sound and the other leading to proofs that are perfectly WI. To construct a non-interactive zap, they let the prover choose the CRS. As the prover could choose a CRS that leads to "unsound" proofs, the prover must actually choose two CRS's that are related in a way that guarantees that at least one of them is of the "sound" type. It must then provide a proof of the statement under both of them. The authors [GOS06a] then show that this protocol still achieves computational WI.

We turn their construction into a proof of knowledge by again doubling the proof, thereby forcing the prover to prove knowledge of a trapdoor which allows to extract the witness from one of the sound proofs. We prove our non-interactive zap of knowledge secure under the same assumptions as Bellare et al.'s S-ZK+SND scheme. Our result is summarized in the following theorem.

Theorem 3.1. *Assuming DLin and DH-KE, there exists a non-interactive zap for circuit satisfiability that satisfies knowledge soundness. The proof size is $O(\lambda k)$, where λ is the security parameter and k is the size of the circuit.*

Let us finally note that our system also implies a proof system which achieves (standard) knowledge soundness, (standard) zero knowledge and *subversion* witness indistinguishability. This is obtained by plugging our zap of knowledge into the construction by Bellare et al. [BFS16] that achieves SND, ZK and S-WI.

Their scheme uses a length-doubling pseudorandom generator (PRG) and a CRS contains a random bit string σ of length 2λ (where λ is the security parameter). A proof for statement x is a zap for the following statement: either x is a valid statement or σ is in the range of the PRG. Using a zap of knowledge (ZaK), knowledge soundness follows from knowledge soundness of the ZaK since with overwhelming probability σ is *not* in the range of the PRG. (The extractor must thus extract a witness for x.) Zero knowledge follows from WI of the zap, as after replacing σ with an element in the range of the PRG, proofs can be simulated using a preimage of σ. Finally, S-WI follows from S-WI of the zap.

Related work. Since the introduction of non-interactive zaps [BOV03, GOS06a], a number of papers have studied and provided different (and more efficient) implementations of zaps. Groth and Sahai [GS08] provided a more general framework for NIWI and NIZK proofs, which leads to more efficient proofs for concrete languages (instead of circuit satisfiability). Furthermore, their proof system can also be based on other assumptions apart from DLin, such as SXDH, allowing for shorter proofs.

Table 3.1: Efficiency and security of the original zaps and our constructions of zaps of knowledge, where w is the number of wires, g the number of gates and $|\mathbb{G}|$ is the size of an element of a group $\mathbb{G}$.

Protocol	Efficiency	Assumptions
Zap [GOS06a]	$(18w + 12g + 5)\,\|\mathbb{G}\|$	DLin
Zap of knowledge, Section 3.4	$(36w + 24g + 14)\,\|\mathbb{G}\|$	DLin, DH-KE
Zap [Ràf15] (of knowledge; Section 3.5)	$(12w + 8g + 3)\,(\|\mathbb{G}_1\|+\|\mathbb{G}_2\|)$	SXDH (ADH-KE)

Bitanski and Paneth [BP15] presented a different approach to constructing zaps and WI proofs based on indistinguishability obfuscation (iO), but constructions using iO are only of theoretical interest. Ràfols [Ràf15] showed how to base non-interactive zaps on Groth-Sahai proofs, thereby achieving an improvement in efficiency (by a constant factor) over the original construction [GOS06a]. Her construction can be implemented in asymmetric ("Type-1") pairing groups.

Her scheme can also serve as the starting point for a scheme achieving knowledge soundness and we explore this in Section 3.5 . (See Table 3.1 for an overview.) Although this scheme is more efficient, we decided to concentrate on building a scheme from [GOS06a], as we can prove it secure under the assumptions that underlie Bellare et al.'s [BFS16] SND+S-ZK scheme; in contrast, a scheme built on asymmetric bilinear groups would require an analogue of the DH-KE assumption in such groups (we refer to it as ADH-KE in Section 3.5). This is a qualitatively different assumption, as without a symmetric pairing it cannot be checked whether the triple returned by the adversary is of the right form (see Fig. 3.1); it would thus not be efficiently decidable if an adversary has won the game. Finally, our main scheme achieves *tight* security, whereas our proof of knowledge soundness in Section 3.5 has a security loss that is linear in the circuit size.

3.1 Cryptographic assumptions

Throughout this chapter, we make use of prime-order abelian groups equipped with a (symmetric) bilinear map. Concretely, we assume the existence of groups $\mathbb{G}, \mathbb{G}_T$ of odd prime order p of length λ and an efficiently computable non-degenerate bilinear map $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. That is, the map e is such that for all $U, V \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p : e(aU, bV) = ab \cdot e(U, V)$, and if U is a generator of $\mathbb{G}$, then $e(U, U)$ is a generator of $\mathbb{G}_T$. We say that a bilinear group is *verifiable* if there exists an efficient verification algorithm that outputs 1 if and only if $\Gamma = (p, \mathbb{G}, \mathbb{G}_T, e)$ is the description of a bilinear group. For instance, the elliptic-curve group of [BBS04] equipped with the Weil pairing is publicly verifiable. In most practical scenarios, the group description is embedded as a part of the protocol specification and agreed upon in advance; in these cases there is no need for verification.

In other words, we assume the existence of a deterministic algorithm GrGen that, given as input the security parameter in unary 1^λ, outputs a bilinear group description Γ. The same assumption was already employed by Bellare et al. [BFS16]. The main advantage in choosing GrGen to be deterministic is that every entity in the scheme can (re)compute the group from the security parameter, and no party must be trusted with generating the group. Moreover, real-world pairing schemes are defined for groups that are fixed for some λ. For the sake of simplicity, we define all our schemes w.r.t. a group description Γ and assume that the security parameter ($\lambda \in \mathbb{N}$ such that $\Gamma := \mathsf{GrGen}(1^\lambda)$) can be derived from Γ.

Our protocol is based on the DH-KE assumption and the existence of a homomorphic extractable commitment scheme. Such schemes have been widely studied and there are constructions from standard assumptions such as the subgroup decision assumption or the decisional linear (DLin) assumption [BBS04]. For this work, we rely on the latter, which is also used in [GOS06a].

Game $\mathrm{DLin}^{b}_{\mathsf{GrGen},\mathcal{A}}(\lambda)$

$\Gamma := (p, \mathbb{G}, \mathbb{G}_T, e, G) := \mathsf{GrGen}(1^\lambda)$
$u, v, r, s \leftarrow\!\!\$\, \mathbb{Z}_p$
if $b = 1$: $H := (r+s)G$
else : $H \leftarrow\!\!\$\, \mathbb{G}$
$b' \leftarrow \mathcal{A}(\Gamma, uG, vG, urG, vsG, H)$
return b'

Game $\text{DH-KE}_{\mathsf{GrGen},\mathcal{A},\mathsf{Ext}}(\lambda)$

$\Gamma := (p, \mathbb{G}, \mathbb{G}_T, e, G) := \mathsf{GrGen}(1^\lambda)$
$r \leftarrow\!\!\$\, \{0,1\}^{\mathcal{A}.\mathsf{rl}(\lambda)}$
$(X, Y, Z) := \mathcal{A}(\Gamma; r)$
$s \leftarrow \mathsf{Ext}(\Gamma, r)$
if $sG = X \vee sG = Y$: **return** 0
return $(e(X, Y) = e(Z, G))$

Figure 3.1: Games for Assumptions 3.2 (DLin) and 3.3 (DH-KE).

The DLin assumption [BBS04] for an abelian group $\mathbb{G} = \langle G \rangle$ of order p states that it is computationally difficult to distinguish $(uG, vG, urG, vsG, (r+s)G)$ with $u, v, r, s \leftarrow\!\!\$\, \mathbb{Z}_p$ from a uniformly random 5-tuple in $\mathbb{G}$.

Assumption 3.2 (DLin)**.** *We say that the Decisional Linear assumption holds for the group generator* GrGen *if for all* PPT *adversaries* $\mathcal{A}$ *we have:*

$$\mathsf{Adv}^{\mathrm{dlin}}_{\mathsf{G},\mathcal{A}}(\lambda) := \left|\Pr\left[\mathrm{DLin}^{0}_{\mathsf{G},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathrm{DLin}^{1}_{\mathsf{G},\mathcal{A}}(\lambda) = 1\right]\right| = \mathsf{negl}(\lambda)\,,$$

where the game $\mathrm{DLin}_{\mathsf{G},\mathcal{A}}(\lambda)$ *is defined in Fig. 3.1.*

The intuition behind DH-KE [BFS16] is that it is difficult for some machine to produce a (Diffie-Hellman) DH triple (xG, yG, xyG) in $\mathbb{G}$ without knowing at least x or y. The assumption is in the spirit of earlier knowledge-of-exponent assumptions [Gro10, BCI$^+$10], whose simplest form states that given $(G, xG) \in \mathbb{G}^2$ it is hard to return (yG, xyG) without knowing y.

Assumption 3.3 (DH-KE)**.** *The Diffie-Hellman Knowledge of Exponent assumption holds for the bilinear group generator* GrGen *if for any* PPT *adversary* $\mathcal{A}$ *there exists a* PPT *extractor* Ext *such that:*

$$\mathsf{Adv}^{\mathrm{dhke}}_{\mathsf{GrGen},\mathcal{A},\mathsf{Ext}}(\lambda) := \Pr\left[\text{DH-KE}_{\mathsf{GrGen},\mathcal{A},\mathsf{Ext}}(\lambda) = 1\right] = \mathsf{negl}(\lambda)\,,$$

where the game $\text{DH-KE}_{\mathsf{GrGen},\mathcal{A},\mathsf{Ext}}(\lambda)$ *is defined in Fig. 3.1.*

In other variants of knowledge of exponent assumptions the adversary is provided with some auxiliary information, which amounts to a stronger assumption. This is typically required as in the security proofs the reduction obtains a challenge which it needs to embed in the input to the adversary. In our specific case, all the proof material is generated by the prover itself, including the CRS. Consequently, the game DH-KE considers an adversary that simply takes as input a group description, without any auxiliary information. Compared to [BFS16], where the adversary is provided with additional information, our variant is thus weaker.

3.2 An extractable commitment scheme from DLin

We recall the homomorphic commitment scheme based on linear encryption [BBS04] by Groth Ostrovsky and Sahai [GOS06a]. It defines two types of key generation: a perfectly hiding and perfectly binding one. Given a bilinear group $\Gamma := (p, \mathbb{G}, \mathbb{G}_T, e, G)$, it defines two key-generation algorithms $\mathsf{Com.G}^{(\mathrm{b})}$ and $\mathsf{Com.G}^{(\mathrm{h})}$ producing binding and hiding keys, respectively:

$\mathsf{Com.G}^{(\mathrm{h})}(\Gamma)$

$\tau \coloneqq (r_u, s_v) \leftarrow\!\!\$\, (\mathbb{Z}_p^*)^2;\ (x, y) \leftarrow\!\!\$\, (\mathbb{Z}_p^*)^2$
$F \coloneqq xG, \quad H \coloneqq yG$
$(U, V, W) \coloneqq (r_u F, s_v H, (r_u + s_v)G)$
$\sigma \coloneqq (F, H, U, V, W)$
return (σ, τ)

$\mathsf{Com.G}^{(\mathrm{b})}(\Gamma)$

$\tau \coloneqq (x, y, z) \leftarrow\!\!\$\, (\mathbb{Z}_p^*)^3;\ (r_u, s_v) \leftarrow\!\!\$\, (\mathbb{Z}_p^*)^2$
$F \coloneqq xG, \quad H \coloneqq yG$
$(U, V, W) \coloneqq (r_u F, s_v H, (r_u + s_v + z)G)$
$\sigma \coloneqq (F, H, U, V, W)$
return (σ, τ)

In order to commit to a value $m \in \mathbb{Z}_p$, one samples $r, s \leftarrow\!\!\$\, \mathbb{Z}_p$ and returns:

$$C = \mathsf{Com.C}(m; r, s) = \big(mU + rF, mV + sH, mW + (r + s)G\big).$$

Since $\mathsf{Com.C}(m_0; r_0, s_0) + \mathsf{Com.C}(m_1; r_1, s_1) = \mathsf{Com.C}(m_0 + m_1; r_0 + r_1, s_0 + s_1)$, commitments are additively homomorphic. A committed value is opened by providing the randomness (r, s). Under a perfectly hiding key, a commitment to m can be opened to any value m', given trapdoor information $\tau = (r_u, s_v)$:

$$\begin{aligned} \mathsf{Com.C}(m; r, s) &= \big((mr_u + r)F, (ms_v + s)V, (mr_u + r + ms_v + s)G\big) \\ &= \mathsf{Com.C}\big(m'; r - (m' - m)r_u, s - (m' - m)s_v\big). \end{aligned} \tag{3.1}$$

Under the DLin assumption, keys output by the perfectly hiding setup are computationally indistinguishable from ones output by the perfectly binding setup. For this reason, the perfectly hiding setup leads to computationally binding commitments and vice versa.

We say that a triple of group elements is *linear* w.r.t. (F, H, G) if it is of the form $(rF, sH, (r + s)G)$ for some $r, s \in \mathbb{Z}_p$. Commitments to 0 are linear triples and every commitment under a *hiding* key is also a linear. Under a *binding* key we have:

$$\mathsf{Com.C}(m; r, s) = \big((mr_u + r)F,\ (ms_v + s)H,\ mzG + (mr_u + r + ms_v + s)G\big).$$

A commitment to m is thus a *linear encryption* [BBS04] of $mzG \in \mathbb{G}_1$ under randomness $(mr_u + r,\ ms_v + s)$. Given a commitment C and the trapdoor information $\tau = (x, y, z)$, one can *extract* the committed message. The extraction algorithm $\mathsf{Com.D}$ is defined as:

$$\mathsf{Com.D}\big(\tau,\ (C_0, C_1, C_2)\big) \coloneqq \mathsf{dLog}\big(z^{-1}(C_2 - x^{-1}C_0 - y^{-1}C_1)\big), \tag{3.2}$$

where dLog can be efficiently computed if the message space is of logarithmic size; for instance, assuming $m \in \{0, 1\}$, we define $\mathsf{Com.D}$ to return 0 if $(C_2 - x^{-1}C_0 - y^{-1}C_1)$ is the identity element, and 1 otherwise.

Theorem 3.4 ([GOS06a])**.** *Assuming DLin,* Com*, as defined above, is an extractable homomorphic commitment scheme that is:*

- *perfectly binding, computationally hiding when instantiated with* $\mathsf{Com.G}^{(\mathrm{b})}$*;*
- *computationally binding, perfectly hiding when instantiated with* $\mathsf{Com.G}^{(\mathrm{h})}$*.*

The "parameter switching" technique, which defines different types of keys that are computationally indistinguishable, has proved very useful and also applies to encryption schemes. The idea has been defined (and named) several times. "Parameter switching" [GOS06a] is also called "meaningful/meaningless encryption" [KN08], "dual-mode encryption" [PVW08] and "lossy encryption" [BHY09].

3.3 Groth-Ostrovsky-Sahai zaps

Proof of binarity

Consider the CRS $\sigma := (F, H, U, V, W)$ and $\Gamma := (p, \mathbb{G}, \mathbb{G}_T, e, G)$ resulting from the execution of (one of the two types of) the key generation algorithm $\mathsf{Com.G}$. Note that F, H are two generators of $\mathbb{G}$ and (U, V, W) is a linear tuple w.r.t. (F, H, G) iff the key generation algorithm is chosen hiding. Groth, Ostrovsky and Sahai [GOS06a] presented a witness-indistinguishable non-interactive proof system Bin for proving that $C \in \mathbb{G}^3$ is a commitment to $\{0, 1\}$ under σ. The intuition behind this construction is that, by the homomorphic property of Com, proving that C commits to a bit is equivalent to showing that either $C = (C_0, C_1, C_2)$ or $C' := C - \mathsf{Com.C}(1; (0, 0)) = C - (U, V, W)$ is a linear tuple with respect to (F, H, G). If we consider the discrete logarithms w.r.t (F, H, G) of the above commitments, i.e. letting $C = (r_0 F, s_0 H, t_0 G)$ and $C' = (r_1 F, s_1 H, t_1 G)$, we have that:

$$\begin{aligned} & C \text{ or } C' \text{ is a linear tuple} \\ \iff & t_0 = r_0 + s_0 \text{ or } t_1 = r_1 + s_1 \\ \iff & (r_0 + s_0 - t_0)(r_1 + s_1 - t_1) = 0 \\ \iff & r_0 r_1 + r_0 s_1 + s_0 r_1 + s_0 s_1 + t_0 t_1 - (r_0 t_1 + t_0 r_1 + s_0 t_1 + t_0 s_1) = 0. \end{aligned} \tag{3.3}$$

Consider a prover $\mathsf{Bin.P}$ holding the witness $(b, r, s) \in \{0, 1\} \times \mathbb{Z}_p \times \mathbb{Z}_p$ for (C, C'), where b indicates which tuple is linear and r, s are its contained randomness. In order to convince a verifier, it proceeds as follows: choose $t \leftarrow\!\!\$\, \mathbb{Z}_p$ and let $\Pi = [\pi_{i,j}]_{i \in \{0,1\}, j \in \{0,1,2\}}$, where:

$$\begin{aligned} \pi_{0,0} &:= r(2b-1)U + r^2 F & \pi_{1,0} &:= s(2b-1)U + (rs+t)F \\ \pi_{0,1} &:= r(2b-1)V + (rs-t)H & \pi_{1,1} &:= s(2b-1)V + s^2 H \\ \pi_{0,2} &:= r(2b-1)W + (r^2+rs+t)G & \pi_{1,2} &:= s(2b-1)W + (s^2+rs-t)G \end{aligned} \tag{3.4}$$

A verifier $\mathsf{Bin.V}$, on input the CRS σ, the statement C and Π, computes $\pi_{2,j} := \pi_{1,j} + \pi_{0,j}$ for $j = 0, 1, 3$ and returns 1 if all the following equations are satisfied:

$$\begin{aligned} e(F, \pi_{0,0}) &= e(C_0, C_0 - U), \\ & \qquad e(F, \pi_{0,1}) + e(H, \pi_{1,0}) = e(C_0, C_1 - V) + e(C_1, C_0 - U) \\ e(H, \pi_{1,1}) &= e(C_1, C_1 - V), \\ & \qquad e(F, \pi_{0,2}) + e(G, \pi_{2,0}) = e(C_0, C_2 - W) + e(C_2, C_0 - U) \\ e(G, \pi_{2,2}) &= e(C_2, C_2 - W), \\ & \qquad e(H, \pi_{1,2}) + e(G, \pi_{2,1}) = e(C_1, C_2 - W) + e(C_2, C_1 - V). \end{aligned} \tag{3.5}$$

If we consider, as we did for the commitments, the discrete logarithms of the proof matrix w.r.t. (F, H, G), i.e. we put

$$m_{i,0} := \log_F(\pi_{i,0}), \qquad m_{i,1} := \log_H(\pi_{i,1}), \qquad m_{i,2} := \log_G(\pi_{i,2}),$$

for $i = 0, 1$, then we observe that the verification equation sets: $m_{2,i} := m_{0,i} + m_{1,i}$, and then checks the following:

$$\begin{aligned} m_{0,0} &= r_0 r_1 & m_{0,1} + m_{1,0} &= r_0 s_1 + s_0 r_1 \\ m_{1,1} &= s_0 s_1 & m_{0,2} + m_{2,0} &= r_0 t_1 + t_0 r_1 \\ m_{2,2} &= t_0 t_1 & m_{1,2} + m_{2,1} &= s_0 t_1 + t_0 s_1 \end{aligned} \tag{3.6}$$

By substitution, this is exactly what Eq. (3.3) affirms.

$\mathsf{Bin.P}(\sigma, C, (b, r, s))$

Construct Π as per Eq. (3.4)
return Π

$\mathsf{Bin.V}(\sigma, C, \Pi)$

$[\pi_{i,j}]_{i\in\{0,1\},j\in\{0,1,2\}} = \Pi$
for $j = 0, 1, 2$ **do** $\pi_{2,j} \coloneqq \pi_{1,j} + \pi_{0,j}$
return (Eq. (3.5))

Figure 3.2: The Bin protocol.

As previously mentioned, the key generation algorithm is identical to Com.G. If the setup is perfectly binding, perfect completeness and perfect soundness follow immediately from Eq. (3.6). Perfect witness indistinguishability follows from the observation that a proof with a witness $(0, r_0, s_0)$ gives the same proof as using witness $(1, r_1, s_1)$ with randomness $t' = t + r_0 s_1 - s_0 r_1$. On the other hand, on a perfectly hiding key generation every commitment is a linear tuple, and thus there is nothing to prove.

Theorem 3.5 ([GOS06a])**.** *The protocol* Bin *is a non-interactive proof system with perfect completeness, perfect soundness, and perfect witness indistinguishability.*

To construct a non-interactive zap (i.e., a WI proof system without a CRS), Groth, Ostrovsky and Sahai [GOS06a] first construct a proof system for circuit satisfiability *with* a CRS, based on the commitment scheme from Section 3.2 and their proof of binarity. Then, in order to make their scheme CRS-less, they define the prover to pick two CRS's that are correlated in a way that makes it impossible for the adversary to cheat under both of them.

As the commitment scheme described in Section 3.2 is homomorphic, it is possible to perform linear operations on commitments, and in particular prove logical relations between them.

First, proving that either C or $C' \coloneqq C - (U, V, W)$ is linear proves that C is a commitment to a bit. In order to prove that committed values satisfy wire assignments of a NAND gate, Groth et al. [GOS06b] observe that if $a, b \in \{0, 1\}$ then $c \coloneqq \neg(a \wedge b)$ iff $t \coloneqq a + b + 2c - 2 \in \{0, 1\}$. Reasoning with homomorphic commitments, we have that three commitments $A \coloneqq (A_0, A_1, A_2)$, $B \coloneqq (B_0, B_1, B_2)$, and $C \coloneqq (C_0, C_1, C_2)$ are bound respectively to the values a, b, c, such that $c = \neg(a \ \wedge \ b)$, if and only if

$$T \coloneqq A + B + 2 \cdot C - 2 \cdot (U, V, W) \tag{3.7}$$

is a commitment to either 0 or 1. Thus, to prove that A, B, C are commitments to values in $\{0, 1\}$ and that C is a commitment to the NAND of the values in A and B, it is sufficient to prove that A, B, C and T are all bit commitments. With these observations, GOS construct a perfectly witness-indistinguishable proof system Circ for circuit satisfiability as follows:

The key generation algorithm Circ.G simply emulates $\mathsf{Com.G}^{(\mathrm{h})}$, that is, it generates a hiding commitment key. The prover $\mathsf{Circ.P}(\sigma, \mathtt{C}, w)$ takes as input a circuit $\mathtt{C}$ and a witness w satisfying

$\mathsf{ZAP.P}(1^\lambda, \phi, w)$

$\Gamma \coloneqq \mathsf{GrGen}(1^\lambda)$; $(\sigma_0, \tau) \leftarrow \mathsf{Circ.G}(\Gamma)$
$\sigma_1 \coloneqq \sigma_0 + (0, 0, 0, 0, G)$
$\pi_0 \leftarrow \mathsf{Circ.P}(\sigma_0, \phi, w)$; $\pi_1 \leftarrow \mathsf{Circ.P}(\sigma_1, \phi, w)$
return (σ_0, π_0, π_1)

$\mathsf{ZAP.V}(\phi, (\sigma_0, \pi_0, \pi_1))$

$\sigma_1 \coloneqq \sigma_0 + (0, 0, 0, 0, G)$
return $\left(\bigwedge_{i\in\{0,1\}} \mathsf{Circ.V}(\sigma_i, \phi, \pi_i)\right)$

Figure 3.3: The (non-interactive) ZAP protocol of [GOS06a].

$\mathtt{C}(w) = 1$, and does the following: represent the circuit evaluation $\mathtt{C}(w)$ in such a way that w_k is the value running in the k-th wire. For each w_k, produce a commitment $C_k \leftarrow \mathsf{Com.C}(\sigma, w_k)$ to w_k and prove it is to a bit under σ using proof system Bin. For each gate, construct T from the commitments corresponding to the ingoing and outgoing wires as above and prove that it too is a commitment to 0 or 1. For the output commitment, create a commitment C_{out} to 1 that can be easily reproduced and checked by the verifier: $C_{\text{out}} \coloneqq \mathsf{Com.C}(\sigma, 1; (0,0))$. Let Π be the collection of all other commitments together with the respective proofs of binarity generated. Return Π.

The verifier $\mathsf{Circ.V}(\sigma, \mathtt{C}, \Pi)$, computes $C_{\text{out}} \coloneqq \mathsf{Com.C}(\sigma, 1; (0,0))$ and for every gate the value T as in Eq. (3.7); using $\mathsf{Bin.V}$, it checks that all the wire commitments are to values in $\{0,1\}$ and respect the gates (by checking the values T); if all verifications succeed, return 1. Otherwise, return 0.

Theorem 3.6 ([GOS06a])**.** *Assuming DLin,* Circ *is a non-interactive, perfectly sound computationally witness-indistinguishable proof system.*

The reason why we cannot let the prover choose the CRS in Circ is that it could chose it as a perfectly hiding CRS and then simulate proofs. However, if the prover must construct two proofs under two different CRS's which are related in such a way that at least one of them is not linear (and thus binding), then the prover cannot cheat. In particular, note that given a 5-tuple $\sigma_0 \in \mathbb{G}^5$, and defining $\sigma_1 \coloneqq \sigma_0 + (0,0,0,0,G)$ then at *most* one of σ_0, σ_1 is linear. At the same time, both of them are valid CRS's. With this last trick, it is straightforward to construct the zap scheme ZAP, as illustrated in Fig. 3.3.

Theorem 3.7 ([GOS06a])**.** *Assuming DLin,* ZAP *is a non-interactive zap with perfect soundness and computational witness indistinguishability.*

Remark 3.8. We note that soundness of ZAP relies only on the fact that Γ is a bilinear group. In [GOS06a] the prover is allowed to generate Γ and it is required that Γ is verifiable. We presented a zap for deterministically generated groups, as considered by Bellare et al. [BFS16], which is also required for our construction of non-interactive zaps of knowledge in the next section.

3.4 ZAK: a non-interactive zap of knowledge

We now present our NIWI argument of knowledge for circuit satisfiability. The high-level idea of our protocol is to double the ZAP proof of [GOS06a] and link the two CRS's so the prover must know the extraction trapdoor for one of them. Whereas the protocol ZAP used two Circ proofs to construct a zap from a proof that requires a CRS, we will use two zap proofs to not only prove circuit satisfiability, but to prove *knowledge* of a satisfying assignment. More specifically, knowledge soundness is obtained by generating two independent zap proofs, and then linking the respective trapdoor information with multiple DH in a matrix of group elements Δ. This additional matrix Δ, that we call *linking element*, is constructed in such a way that (under DH-KE) it is possible to recover the trapdoor from one of the two zap proofs, and use it to extract the witness from the commitments contained in a valid zap proof. Witness indistinguishability of the single proofs follows immediately from [GOS06a], but our proofs also contain the linking element Δ, which depend on the randomness of the CRS's. We thus have to argue that these additional elements do not harm witness indistinguishability.

Bellare et al. [BFS16] also used an extractor to recover the trapdoor hidden in an adversarially generated CRS to construct a scheme satisfying subversion-zero knowledge. Our protocol is detailed in Fig. 3.4, where by DH we denote the algorithm that checks that $\delta_{i,j}$ is the CDH of $(\sigma_{0,0})_i$ and $(\sigma_{1,0})_j$ (see below).

$\mathsf{ZAK.P}(1^\lambda, \phi, w)$

$\Gamma \coloneqq \mathsf{GrGen}(1^\lambda)$
for $i = 0, 1$ **do**
  $(\sigma_{i,0}, \tau_i) \leftarrow \mathsf{Circ.G}(\Gamma)$
  $\sigma_{i,1} \coloneqq \sigma_{i,0} + (0, 0, 0, 0, G)$
  $\pi_{i,0} \leftarrow \mathsf{Circ.P}(\sigma_{i,0}, \phi, w)$
  $\pi_{i,1} \leftarrow \mathsf{Circ.P}(\sigma_{i,1}, \phi, w)$
Compute Δ from τ_0, τ_1 as in Eq. (3.8).
$\Sigma \coloneqq [\sigma_{i,0}]_{i\in\{0,1\}}$, $\Pi = [\pi_{i,j}]_{i,j\in\{0,1\}}$
return (Σ, Δ, Π)

$\mathsf{ZAK.V}(\phi, (\Sigma, \Delta, \Pi))$

// Check if Δ is consistent with Σ
if not $\mathsf{DH}(\Delta, \Sigma)$: **return** 0
for i **in** $\{0, 1\}$ **do**
  $\sigma_{i,1} \coloneqq \sigma_0 + (0, 0, 0, 0, G)$
return $\big(\ \bigwedge_{i,j\in\{0,1\}} \mathsf{Circ.V}(\sigma_{i,j}, \phi, \pi_{i,j})\ \big)$

Figure 3.4: The ZAK protocol.

The trapdoor information $\tau_0 = (x_0, y_0)$ and $\tau_1 = (x_1, y_1)$ is correlated in Δ to form the following products:

$$\Delta \coloneqq [\delta_{i,j}]_{i,j\in\{0,1\}} = \begin{bmatrix} x_0x_1G & x_0y_1G \\ y_0x_1G & y_0y_1G \end{bmatrix} \tag{3.8}$$

Correctness of Δ can be checked by the verification algorithm using the bilinear map. For $i = 0, 1$, let the CRS be $\sigma_i = (F_i, H_i, U_i, V_i, W_i)$, and let x_i, y_i be such that:

$$F_i \coloneqq x_iG, \qquad H_i \coloneqq y_iG,$$

in which case Δ is constructed as in Eq. (3.8). The verifier checks that the following holds:

$$\begin{aligned} e(\delta_{0,0}, G) = e(F_0, F_1), \quad e(\delta_{0,1}, G) = e(F_0, H_1), \\ e(\delta_{1,0}, G) = e(H_0, F_1), \quad e(\delta_{1,1}, G) = e(H_0, H_1). \end{aligned} \tag{3.9}$$

Let us denote by DH the algorithm that, given as input Σ and Δ returns 1 if all equalities of Eq. (3.9) are satisfied, and 0 otherwise. This procedure is used by the verification equation, as detailed in Fig. 3.4.

We now proceed with the proof of our main result, Theorem 3.1, which we rephrase here for completeness:

Theorem 3.1. *Assume that DLin and DH-KE hold for* GrGen. *Then* ZAK *as defined in Fig. 3.4 is a non-interactive zap that satisfies knowledge soundness and witness indistinguishability. In particular, we have*

$$\mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{ZAK}}(\lambda) \le 4 \cdot \mathsf{Adv}^{\text{dh-ke}}(\lambda) \quad \textit{and} \quad \mathsf{Adv}^{\mathrm{wi}}_{\mathsf{ZAK}}(\lambda) \le 8 \cdot \mathsf{Adv}^{\mathrm{dlin}}(\lambda).$$

Completeness of the protocol is trivial: the prover (respectively, the verifier) simply performs 4 iterations of Circ proofs (respectively, verifications), and therefore correctness is implied by Theorem 3.6 and the fact that Δ as in Eq. (3.8) satisfies Eq. (3.9). We now prove knowledge soundness and witness indistinguishability.

Proof (computational knowledge soundness). We show that for any adversary able to produce a valid proof we can construct a PPT extractor that can extract a witness from such a proof with overwhelming probability.

Game $\mathrm{KSND}_{\mathsf{ZAK},\text{CIRC-SAT},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)$

$\Gamma \coloneqq \mathsf{GrGen}(1^\lambda)\,;\ r \leftarrow\!\!\$\ \{0,1\}^{\mathcal{A}.\mathsf{rl}(\lambda)}$

$(\mathtt{C}, (\Sigma, \Delta, \Pi)) \coloneqq \mathcal{A}(1^\lambda; r)$

$w \leftarrow \mathsf{Ext}(1^\lambda, r)$

return $\mathsf{ZAK.V}(\mathtt{C}, (\Sigma, \Delta, \Pi))$ **and** $\mathtt{C}(w) \neq 1$

Figure 3.5: Knowledge soundness game for the ZAK protocol.

Let $\mathcal{A}$ be an adversarial prover in game $\mathrm{KSND}_{\Pi,\mathcal{A}}(\lambda)$ (Fig. 2.2, with $\Pi.\mathsf{G}$ void). On input 1^λ, $\mathcal{A}$ returns a proof consisting of $\sigma_{i,0} = (F_i, H_i, U_i, V_i, W_i)$ for $i \in \{0,1\}$, of $\Delta = [\delta_{i,j}]_{i,j\in\{0,1\}}$ and $\Pi = [\pi_{i,j}]_{i,j\in\{0,1\}}$. The game $\mathrm{KSND}_{\mathsf{ZAK},\text{CIRC-SAT}}(\lambda)$ is given in Fig. 3.5. From $\mathcal{A}$ we construct four adversaries $\mathcal{A}_{i,j}$ (for $i,j \in \{0,1\}$) that execute $\mathcal{A}$ and output some components of the proof produced by $\mathcal{A}$, namely

$$
\begin{aligned}
(F_0,\, F_1,\, \delta_{0,0}) &= (x_0G,\, x_1G,\, x_0x_1G), && \text{(for } \mathcal{A}_{0,0}\text{)}\\
(F_0,\, H_1,\, \delta_{0,1}) &= (x_0G,\, y_1G,\, x_0y_1G), && \text{(for } \mathcal{A}_{0,1}\text{)}\\
(H_0,\, F_1,\, \delta_{1,0}) &= (y_0G,\, x_1G,\, y_0x_1G), && \text{(for } \mathcal{A}_{1,0}\text{)}\\
(H_0,\, H_1,\, \delta_{0,1}) &= (y_0G,\, y_1G,\, y_0y_1G), && \text{(for } \mathcal{A}_{1,1}\text{)}
\end{aligned}
$$

where x_i, y_i are such that $F_i = x_iG$, $H_i = y_iG$, and these four equations hold if $\mathsf{ZAK.V}(\mathtt{C}, (\Sigma, \Delta, \Pi))$ returns 1. By the DH-KE assumption there exist extractors $\mathsf{Ext}_{i,j}$ for each of the adversaries $\mathcal{A}_{i,j}$ that given its coins outputs:

$$
\begin{aligned}
&x_0 \text{ or } x_1, && x_0 \text{ or } y_1, && \text{(for } \mathsf{Ext}_{0,0},\, \mathsf{Ext}_{0,1}\text{)}\\
&y_0 \text{ or } x_1, && y_0 \text{ or } y_1 && \text{(for } \mathsf{Ext}_{1,0},\, \mathsf{Ext}_{1,1}\text{)}
\end{aligned}
$$

if the above equations hold. The statement $(x_0 \vee x_1) \wedge (y_0 \vee x_1) \wedge (x_0 \vee y_1) \wedge (y_0 \vee y_1)$ is logically equivalent to $(x_0 \wedge y_0) \vee (x_1 \wedge y_1)$. This means that together, these four extractors allow to recover either (x_0, y_0) or (x_1, y_1), that is, the extraction trapdoor for one of the CRS's. Let i^* be such that (x_{i^*}, y_{i^*}) is the extracted pair.

For $j \in \{0,1\}$, let $F_{i^*}, H_{i^*}, U_{i^*}, V_{i^*}, W_{i^*} \in \mathbb{G}$ be such that $\sigma_{i^*,j} = (F_{i^*}, H_{i^*}, U_{i^*}, V_{i^*}, W_{i^*} + jG)$. Let $j^* \in \{0,1\}$ be the smallest integer satisfying:

$$x_{i^*}^{-1}U_{i^*} + y_{i^*}^{-1}V_{i^*} - (W_{i^*} + j^*G) \neq 0G.$$

The above implies that σ_{i^*,j^*} is not a linear tuple, which means that it is a binding CRS. Let $C_{(i^*,j^*),k}$ denote the commitment to the k-th wire contained in π_{i^*,j^*}. Using the extraction algorithm described in Eq. (3.2) we can recover this witness:

$$w_k = \mathsf{Com.D}\big((x_{i^*}, y_{i^*}),\ C_{(i^*,j^*),k}\big).$$

It remains to prove that the extracted witness is indeed correct. Upon receiving a valid proof from adversary $\mathcal{A}$, we know from the verification equation (the subroutine DH) that each $\mathcal{A}_{i,j}$ will output a DH triple. Therefore, extractors $\mathsf{Ext}_{i,j}$ together recover $\tau_{i^*} = (x_{i^*}, y_{j^*})$ with probability at least $1 - \sum_{i,j\in\{0,1\}} \mathsf{Adv}^{\mathrm{dhke}}_{\mathsf{GrGen},\mathcal{A}_{i,j},\mathsf{Ext}_{i,j}}(\lambda)$, that is, by DH-KE, with overwhelming probability. Since the commitment scheme Com is perfectly binding if the CRS is not a linear tuple (Theorem 3.4), a message w_k is always successfully extracted. Correctness of w_k follows from the underlying proof system: by perfect soundness of Bin (Theorem 3.5) we are guaranteed that $w_k \in \{0,1\}$; by perfect soundness of Circ (Theorem 3.6) that each gate evaluation is correct. The bound in the construction of the extractor is tight: we have $\mathsf{Adv}^{\mathrm{ksnd}}(\lambda) \leq 4 \cdot \mathsf{Adv}^{\mathrm{dhke}}(\lambda)$. □

Oracle PROVE in H_1, $\boxed{\mathsf{H}_2}$, and $\boxed{\mathsf{H}_3}$

$\Gamma \coloneqq \mathsf{GrGen}(1^\lambda)$
$(\sigma_{0,0}, \tau_i) \leftarrow \mathsf{Circ.G}(\Gamma)$
$\boxed{(\sigma_{0,0}, \tau_i) \leftarrow \mathsf{Com.G}^{(\mathrm{b})}(\Gamma)}$
$\sigma_{0,1} \coloneqq \sigma_{0,0} + (0,0,0,0,G)$
$\boxed{\begin{array}{l}(\sigma_{0,1}, \tau_i) \leftarrow \mathsf{Circ.G}(\Gamma)\\ \sigma_{0,0} \coloneqq \sigma_{0,1} - (0,0,0,0,G)\end{array}}$
$\pi_{0,0} \leftarrow \mathsf{Circ.P}(\sigma_{0,0}, \mathsf{C}, w_1)$
$\pi_{0,1} \leftarrow \mathsf{Circ.P}(\sigma_{0,1}, \mathsf{C}, w_0)$
// The second zap is as in ZAK.P using w_0.
$(\sigma_{1,0}, \pi_{1,0}, \pi_{1,1}) \leftarrow \mathsf{ZAP.P}(1^\lambda, \mathsf{C}, w_0)$
Compute Δ as in Eq. (3.8).
return (Σ, Δ, Π)

Oracle PROVE in H_4 and $\boxed{\mathsf{H}_5}$

$\Gamma \coloneqq \mathsf{GrGen}(1^\lambda)$
$(\sigma_{0,1}, \tau_i) \leftarrow \mathsf{Circ.G}(\Gamma)$
$\sigma_{0,0} \coloneqq \sigma_{0,1} - (0,0,0,0,G)$
$\boxed{(\sigma_{0,1}, \tau_i) \leftarrow \mathsf{Com.G}^{(\mathrm{b})}(\Gamma)}$
$\pi_{0,0} \leftarrow \mathsf{Circ.P}(\sigma_{0,0}, \mathsf{C}, w_1)$
$\pi_{0,1} \leftarrow \mathsf{Circ.P}(\sigma_{0,1}, \mathsf{C}, w_1)$
// The second zap is as in ZAK.P using w_0.
$(\sigma_{1,0}, \pi_{1,0}, \pi_{1,1}) \leftarrow \mathsf{ZAP.P}(1^\lambda, \mathsf{C}, w_0)$
Compute Δ as in Eq. (3.8).
return (Σ, Δ, Π)

Figure 3.6: Overview of the simulations of the prove oracle in the first hybrid games for the proof of WI. Hybrids H_1 and H_4 are defined by ignoring all boxes (the light gray highlights the differences with respect to the previous hybrids), whereas $\boxed{\mathsf{H}_2}$ and $\boxed{\mathsf{H}_5}$ include the light boxes but not the gray one and $\boxed{\mathsf{H}_3}$ includes all boxes.

Proof (computational witness indistinguishability). Consider an adversary in the WI game (Fig. 2.3, where $\Pi.\mathsf{G}$ is void) that makes $q = q(\lambda)$ queries to the PROVE oracle, each of the form $(\mathsf{C}^{(k)}, w_0^{(k)}, w_1^{(k)})$, for $0 \le k < q$. Consider the following sequence of hybrid games where H_0 corresponds to $\mathrm{WI}^0_{\mathsf{ZAK},\text{CIRC-SAT},\mathcal{A}}(1^\lambda)$ and H_{12} corresponds to $\mathrm{WI}^1_{\mathsf{ZAK},\text{CIRC-SAT},\mathcal{A}}(1^\lambda)$. The games differ in how the PROVE oracle is implemented, which is specified in Fig. 3.6 for the first half of the hybrids (the second half is analogous). We give an overview of all hybrids in Table 3.2 below.

H_0 The challenger simulates an honest PROVE oracle, using (for every $k < q$) the first witness $w_0^{(k)}$ supplied by the adversary. It outputs $(\Sigma^{(k)}, \Delta^{(k)}, \Pi^{(k)})$, where in particular we recall:

$$\Sigma^{(k)} = \begin{bmatrix} \sigma_{0,0}^{(k)} = (F_0^{(k)}, H_0^{(k)}, U_0^{(k)}, V_0^{(k)}, W_0^{(k)}) \\ \sigma_{1,0}^{(k)} = (F_1^{(k)}, H_1^{(k)}, U_1^{(k)}, V_1^{(k)}, W_1^{(k)}) \end{bmatrix} \quad \text{and} \quad \Pi^{(k)} = \begin{bmatrix} \pi_{0,0}^{(k)} & \pi_{0,1}^{(k)} \\ \pi_{1,0}^{(k)} & \pi_{1,1}^{(k)} \end{bmatrix}.$$

Recall that the two rows of $[\Sigma^{(k)}|\Pi^{(k)}]$ are independent zaps and that $\sigma_{0,0}^{(k)}$ and $\sigma_{1,0}^{(k)}$ are chosen to be *hiding*. The PROVE oracle computes $\sigma_{i,j}^{(k)}$ which is of the form $\sigma_{i,j}^{(k)} = (F_i^{(k)}, H_i^{(k)}, U_i^{(k)}, V_i^{(k)}, W_i^{(k)} + jG)$, for $i, j \in \{0,1\}$. Furthermore, $\pi_{i,j}^{(k)}$ is a Circ proof using $w_0^{(k)}$ under the CRS $\sigma_{i,j}^{(k)}$.

H_1 For every PROVE query, the simulator uses witness $w_1^{(k)}$ (instead of $w_0^{(k)}$) to produce $\pi_{0,0}^{(k)}$. As the respective CRS $\sigma_{0,0}^{(k)}$ was generated using the perfectly hiding commitment setup $\mathsf{Circ.G}$, the two hybrids are distributed equivalently (any commitment under a hiding key is a random linear triple; cf. Eq. (3.1)).

H_2 For every PROVE query, the simulator now generates CRS $\sigma_{0,0}^{(k)}$ as a *binding* key via $\mathsf{Com.G}^{(b)}$; $\sigma_{0,1}^{(k)}$ is generated as before (adding $(0,0,0,0,G)$), and so are all proofs. Note that the linking

Table 3.2: Overview of changes throughout the hybrids: (h) denotes hiding setup; (b) denotes binding setup; w_b identifies the witness used to produce the proof.

Hybrid	$\sigma_{0,0}^{(k)}$	$\pi_{0,0}^{(k)}$	$\sigma_{0,1}^{(k)}$	$\pi_{0,1}^{(k)}$	$\sigma_{1,0}^{(k)}$	$\pi_{1,0}^{(k)}$	$\sigma_{1,1}^{(k)}$	$\pi_{1,1}^{(k)}$
H_0	(h)	w_0	(b)	w_0	(h)	w_0	(b)	w_0
H_1		w_1						
H_2	(b)							
H_3			(h)					
H_4				w_1				
H_5			(b)					
H_6	(h)							
H_7						w_1		
H_8					(b)			
H_9							(h)	
H_{10}								w_1
H_{11}							(b)	
H_{12}	(h)	w_1	(b)	w_1	(h)	w_1	(b)	w_1

elements $\Delta^{(k)}$ can be constructed knowing only the trapdoor $(x_1^{(k)}, y_1^{(k)})$ of the CRS $\sigma_{1,0}^{(k)}$, which remained unchanged:

$$\Delta^{(k)} = \begin{bmatrix} y_1^{(k)} H_0^{(k)} & y_1^{(k)} F_0^{(k)} \\ x_1^{(k)} H_0^{(k)} & x_1^{(k)} F_0^{(k)} \end{bmatrix}. \tag{3.10}$$

H_1 and H_2 are computationally indistinguishable under the DLin assumption: given a DLin challenge (F, H, U, V, W), the reduction can exploit the random self-reducibility property of DLin to construct q instances of the DLin challenge: $\forall k < q$ select $\bar{x}^{(k)}, \bar{y}^{(k)}, \bar{r}^{(k)}, \bar{s}^{(k)}, \bar{z}^{(k)} \leftarrow\!\!\$\, \mathbb{Z}_p$ and compute $\sigma_{0,0}^{(k)}$ as $(\bar{x}^{(k)}F,\ \bar{y}^{(k)}H,\ \bar{r}^{(k)}\bar{x}^{(k)}F + \bar{z}^{(k)}\bar{x}^{(k)}U,\ \bar{s}^{(k)}\bar{y}^{(k)}H + \bar{z}^{(k)}\bar{y}^{(k)}V,\ (\bar{r}^{(k)} + \bar{s}^{(k)})G + \bar{z}^{(k)}W)$.

Each $\sigma_{0,0}^{(k)}$ is a random linear tuple if and only if the DLin challenge is, and it is a uniformly random tuple if the DLin challenge is, as shown in [BFS16]. Computing $\sigma_{1,0}^{(k)}$ as in H_1 (hiding) and defining Δ as in Eq. (3.10), the simulator generates the rest of the game as defined. It returns the adversary's guess and thus breaks DLin whenever the adversary distinguishes H_1 and H_2.

H_3 The simulator replaces each CRS $\sigma_{0,1}^{(k)}$ for all $k < q$ with a *hiding* commitment and defines $\sigma_{0,0}^{(k)} \coloneqq \sigma_{0,1}^{(k)} - (0, 0, 0, 0, G)$, which is therefore (once again) binding. More specifically, the simulator creates a linear tuple invoking Circ.G:

$$\sigma_{0,1}^{(k)} = (x_0^{(k)}G,\ y_0^{(k)}G,\ x_0^{(k)}r^{(k)}G,\ y_0^{(k)}s^{(k)}G,\ (r^{(k)} + s^{(k)})G)$$

where $x_0^{(k)}, y_0^{(k)}, r^{(k)}, s^{(k)} \leftarrow\!\!\$\, \mathbb{Z}_p$.

The two distributions are proven computationally indistinguishable under DLin by an argument analogous to the one for $\mathsf{H}_1 \to \mathsf{H}_2$. This time the challenger constructs all the

Game $\mathrm{XDH}^b_{I,\mathsf{GrGen},\mathcal{A}}(\lambda)$

$\Gamma := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2) := \mathsf{GrGen}(1^\lambda)$
$x, y \leftarrow\!\$\, \mathbb{Z}_p^*$
if $b = 1:\quad H := xyG_I$
else : $H \leftarrow\!\$\, \mathbb{G}_I$
$b' \leftarrow \mathcal{A}(\Gamma, xG_I, yG_I, H)$
return b'

Game $\text{ADH-KE}_{\mathsf{GrGen},\mathcal{A},\mathsf{Ext}}(\lambda)$

$\Gamma := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2) := \mathsf{GrGen}(1^\lambda)$
$r \leftarrow\!\$\, \{0,1\}^{\mathcal{A}.\mathsf{rl}(\lambda)}$
$(X, Y, Z) := \mathcal{A}(\Gamma; r)$
$s \leftarrow \mathsf{Ext}(\Gamma, r)$
if $sG_1 = X \vee sG_1 = Y:$ **return** 0
return $(Z = \log_{G_1}(X) \cdot Y)$

Figure 3.7: Games for Assumptions 3.2 (SXDH) and 3.3 (ADH-KE).

instances of the DLin challenge for $\sigma^{(k)}_{0,1}$, while $\sigma^{(k)}_{0,0}$ is derived. From there, the proof proceeds identically.

H_4 The simulator replaces each proof $\pi^{(k)}_{0,1}$ by using $w^{(k)}_1$ instead of $w^{(k)}_0$ $(\forall k < q)$.

This hybrid is equivalently distributed as the previous one; this is proved via the same argument as for $\mathsf{H}_0 \to \mathsf{H}_1$.

H_5 The simulator switches $\sigma^{(k)}_{0,1}$ from a hiding to a binding key. This game hop is analogous to the hop $\mathsf{H}_1 \to \mathsf{H}_2$ (which switched $\sigma^{(k)}_{0,0}$ from hiding to binding).

H_6 The simulator switches $\sigma^{(k)}_{0,0}$ from binding to hiding. Indistinguishability from the previous hybrid is shown analogously to the hop $\mathsf{H}_2 \to \mathsf{H}_3$. Note that in this hybrid the first zap $(\sigma^{(k)}_{0,0}, \pi^{(k)}_{0,0}, \pi^{(k)}_{0,1})$ is distributed according to the protocol specification, but using witness $w^{(k)}_1$.

Hybrids H_7 to H_{12} are now defined analogously to hybrids H_1 to H_6, except for applying all changes to $\sigma^{(k)}_1$ and $\pi^{(k)}_{1,0}$ and $\pi^{(k)}_{1,1}$. In hybrid H_{12} the adversary is then given arguments of knowledge for witness w_1.

As the difference between hybrids H_1 and H_{12} is bounded by 8 times the advantage of a DLin distinguisher, the adversary has total advantage

$$\mathsf{Adv}^{\mathrm{wi}}_{\mathsf{ZAK},\mathsf{C},\mathcal{A}}(\lambda) \leq 8 \cdot \mathsf{Adv}^{\mathrm{dlin}}_{\mathsf{ZAK},\mathsf{C},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)\,.$$

The bound is thus tight. □

3.5 Non-interactive zaps of knowledge in asymmetric groups

In this section we show an alternative and more efficient approach to constructing non-interactive zaps of knowledge for circuit satisfiability. In contrast to symmetric bilinear groups used in the previous section, we will work with asymmetric pairings, that is, bilinear maps $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ (where $\mathbb{G}_1 = \langle G_1 \rangle$, $\mathbb{G}_2 = \langle G_2 \rangle$ and $\mathbb{G}_T$ are abelian additive groups of prime order p). We assume a deterministic algorithm GrGen that outputs an (asymmetric) group description $\Gamma := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$.

Cryptographic assumptions

By extending GOS proofs [GOS06b], Goth and Sahai [GS08] provide a general framework for non-interactive witness-indistinguishable (NIWI) proof systems, which can be based (among other computational assumptions) on SXDH. The SXDH assumption for an asymmetric pairing group

generator GrGen informally states that the decisional Diffie-Hellman assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

Assumption 3.2 (SXDH)**.** *We say that the Symmetric External Diffie-Hellman assumption holds for the asymmetric bilinear group generator* GrGen *if for all* PPT *adversaries* $\mathcal{A}$ *we have:*

$$\mathsf{Adv}^{\mathrm{XDH1}}_{\mathsf{GrGen},\mathcal{A}}(\lambda) \coloneqq \left|\Pr\left[\mathrm{XDH}^0_{1,\mathsf{GrGen},\mathcal{A}}(\lambda)=1\right] - \Pr\left[\mathrm{XDH}^1_{1,\mathsf{GrGen},\mathcal{A}}(\lambda)=1\right]\right| = \mathsf{negl}(\lambda) \text{ and}$$
$$\mathsf{Adv}^{\mathrm{XDH2}}_{\mathsf{GrGen},\mathcal{A}}(\lambda) \coloneqq \left|\Pr\left[\mathrm{XDH}^0_{2,\mathsf{GrGen},\mathcal{A}}(\lambda)=1\right] - \Pr\left[\mathrm{XDH}^1_{2,\mathsf{GrGen},\mathcal{A}}(\lambda)=1\right]\right| = \mathsf{negl}(\lambda)\,,$$

where $\mathrm{XDH}^b_{I,\mathsf{GrGen},\mathcal{A}}(\lambda)$ *(for* $I=1,2$*) is defined in Fig. 3.7.*

In order to construct zaps of knowledge over asymmetric bilinear groups, we require the analogue of DH-KE for such groups, in particular for their first base group $\mathbb{G}_1$. We give a formal definition.

Assumption 3.3 (ADH-KE)**.** *The Asymmetric Diffie-Hellman Knowledge of Exponent assumption holds for (the first base group of) the asymmetric group generator* GrGen *if for any* PPT *adversary* $\mathcal{A}$ *there exists a* PPT *extractor* Ext *such that:*

$$\mathsf{Adv}^{\text{adh-ke}}_{\mathsf{GrGen},\mathcal{A},\mathsf{Ext}}(\lambda) \coloneqq \Pr\left[\text{ADH-KE}_{\mathsf{GrGen},\mathcal{A},\mathsf{Ext}}(\lambda)\right] = \mathsf{negl}(\lambda)\,,$$

where the game $\text{ADH-KE}_{\mathsf{GrGen},\mathcal{A},\mathsf{Ext}}(\lambda)$ *is defined in Fig. 3.7.*

Groth-Sahai (GS) proofs [GS08] achieve improved efficiency by working for group-dependent languages, in contrast to the more elementary proof system Bin of "bit commitment" (given in Section 3.3) used for circuit satisfiability. More recently, Ràfols [Ràf15] gave a construction of non-interactive zaps from GS proofs, which leads to more efficient non-interactive zaps (by a constant factor). Relying on the asymmetric variant of the DH-KE assumption, we show how to achieve knowledge soundness also for GS zaps. Interestingly, the scheme does not require any alteration to the protocol, that is, under ADH-KE we can show that a GS zap is already a GS zap of knowledge.

Groth-Sahai zaps

We first describe the GS-based zap and then argue that it satisfies knowledge soundness under ADH-KE.

SXDH commitments and proofs of binarity. The SXDH commitment scheme of Groth and Sahai [GS08] allows to commit to values in $\mathbb{Z}_p$ both in $\mathbb{G}_1$ and in $\mathbb{G}_2$ (here we parametrize the algorithm with $I \in \{1,2\}$ for compactness). The properties of the scheme are very similar to those of GOS's [GOS06a] DLin-based commitments (Section 3.2). Again, commitment keys can be generated in one of two possible "modes", one perfectly hiding and one perfectly binding.

$\underline{\mathsf{Com.G}^{(\mathrm{h})}_I(\Gamma)}$

$\tau \coloneqq (x,y) \leftarrow\!\$\, (\mathbb{Z}_p^*)^2$
$\mathbf{V} \coloneqq (xG_I, G_I)^\top$
$\mathbf{W} \coloneqq (xyG_I, yG_I)^\top$
$\sigma \coloneqq (\mathbf{V},\mathbf{W})$
return (σ,τ)

$\underline{\mathsf{Com.G}^{(\mathrm{b})}_I(\Gamma)}$

$\tau \coloneqq (x,z) \leftarrow\!\$\, (\mathbb{Z}_p^*)^2;\ y \leftarrow\!\$\, \mathbb{Z}_p^*$
$\mathbf{V} \coloneqq (xG_I, G_I)^\top$
$\mathbf{W} \coloneqq (xyG_I, (y+z)G_I)^\top$
$\sigma \coloneqq (\mathbf{V},\mathbf{W})$
return (σ,τ)

The commitment key thus consists of vectors $\mathbf{V}, \mathbf{W} \in \mathbb{G}_I^2$, for $I = 1, 2$. Committing to a value $m \in \mathbb{Z}_p$ is performed by sampling $r \leftarrow\!\!\$\, \mathbb{Z}_p$ and computing:

$$\mathsf{Com.C}_I(m; r) \coloneqq m\mathbf{W} + r\mathbf{V}.$$

The commitment scheme is additively homomorphic, since $\mathsf{Com.C}_I(m_0; r_0) + \mathsf{Com.C}_I(m_1; r_1) = \mathsf{Com.C}_I(m_0 + m_1; r_0 + r_1)$. The two setups $\mathsf{Com.G}_I^{(\mathrm{h})}$ and $\mathsf{Com.G}_I^{(\mathrm{b})}$ are computationally indistinguishable under DDH in $\mathbb{G}_I$: hiding setup returns a DH triple (V_0, W_1, W_0) with respect to $V_1 = G_I$, whereas binding setup returns random values (V_0, W_1, W_0).

If $\mathbf{V}, \mathbf{W}$ are linearly dependent, which is the case when generated by $\mathsf{Com.G}_I^{(\mathrm{h})}$, then the commitment is perfectly hiding; a commitment $\mathbf{C}$ to a value m can be opened to any value $m' \in \mathbb{Z}_p$ given the trapdoor information $\tau = (x, y)$:

$$\mathsf{Com.C}_I(m; r) = \begin{pmatrix} x(my + r)G_I \\ (my + r)G_I \end{pmatrix} = \mathsf{Com.C}_I(m'; r + (m - m')y).$$

If $\mathbf{V}, \mathbf{W}$ are linearly independent then the commitment is perfectly binding and for message spaces of logarithmic size the committed value can be *extracted* using the trapdoor information $\tau = (x, z)$ generated by $\mathsf{Com.G}_I^{(\mathrm{b})}$:

$$m = \mathsf{Com.E}_I(\tau,\ \mathbf{C}) \coloneqq \mathsf{dLog}(z^{-1}(C_1 - x^{-1}C_0)).$$

A commitment in $\mathbb{G}_I$ can be shown to be bound to a bit via two quadratic equations in $\mathbb{Z}_p$, as introduced by Groth and Sahai [GS08]. To do so, we require another commitment in the opposite source group $\mathbb{G}_{3-I}$. Let b_1 be the value committed over $\mathbb{G}_1$ and b_2 the value committed over $\mathbb{G}_2$. Our goal is to prove that $b_1 \in \{0, 1\}$; at the same time we prove $b_1 = b_2$. This can be done by proving that the commitments satisfy the following two equations:

$$b_1(b_2 - 1) = 0 \qquad \text{and} \qquad b_2(b_1 - 1) = 0\ . \tag{3.11}$$

We refer the reader to [GS08, §9 p. 28] for how to construct proofs for the above equations being satisfied by the committed values.[1] A proof for one such equation consists of one element from each source group. We can thus define a proof system Bin, which, given a commitment in $\mathbb{G}_1$ and another one in $\mathbb{G}_2$, proves that the committed values are bits using $2(|\mathbb{G}_1| + |\mathbb{G}_2|)$ group elements. The key generation algorithm $\mathsf{Bin.G}$ simply executes $\mathsf{Com.G}_1^{(b)}$ and $\mathsf{Com.G}_2^{(b)}$.

Proofs of circuit satisfiability. Now that we have a witness-indistinguishable system for proving that a commitment is bound to a bit $b \in \{0, 1\}$ over asymmetric bilinear groups under the SXDH assumption, we can construct a protocol Circ for proving circuit satisfiability analogously to scheme by GOS [GOS06b] given in Section 3.3: The prover commits to each wire in the circuit twice (once in $\mathbb{G}_1$ and once in $\mathbb{G}_2$), proves that the committed values are to either 0 or 1, and for each NAND gate with input wire values a, b and output wire value c it proves that $(a+b+2c-2) \in \{0, 1\}$. The output commitment is fixed again to $\mathsf{Com.C}(1; 0)$. This defines $\mathsf{Circ.P}((\sigma_1, \sigma_2), \phi, w)$ where σ_I is a CRS in group $\mathbb{G}_I$, ϕ is the statement, i.e., a circuit description and w is the witness, a satisfying assignment. A proof π consists thus of commitments $C_{1,k} \in \mathbb{G}_1^2$ and $C_{2,k} \in \mathbb{G}_2^2$ and proofs of binarity π_k for every wire w_k, and moreover proofs π_i for every gate g_i.

[1] In GS notation, the equations are defined by setting the parameters $\mathbf{a} \coloneqq (0)$, $\mathbf{b} \coloneqq (-1)$, $\Gamma \coloneqq [1]$, and $t \coloneqq 0$ for the first equation and $\mathbf{a} \coloneqq (-1)$, $\mathbf{b} \coloneqq (0)$, $\Gamma \coloneqq [1]$, $t \coloneqq 0$ for the second.

A zap from SXDH. Again, a ZAP is constructed from Circ analogously to the GOS zap [GOS06a] given in Section 3.3. There, a zap consisted of 2 Circ-proofs for two related CRS's of which one was guaranteed to be binding and thus lead to sound proofs. For SXDH we now create two related CRS's in *each* group $\mathbb{G}_1$ and $\mathbb{G}_2$, so we are guaranteed that for each group one of them is binding. Intuitively, we need to construct 4 Circ proofs, one for each combination of a CRS in $\mathbb{G}_1$ with one from $\mathbb{G}_2$. We are then guaranteed that one of the four proofs is under *two* binding CRS, which asserts that the prover cannot cheat. (Note that we do not actually need four full Circ proofs, as they can share the commitments.)

More specifically, the prover constructs two CRS's $\sigma_{I,0}$, for $I = 1, 2$, for perfectly hiding SXDH commitments in $\mathbb{G}_1$ and $\mathbb{G}_2$. Then, it computes (again for $I = 1, 2$):

$$\sigma_{I,1} \coloneqq \sigma_{I,0} - ((0,0)^\top, (0, G_I)^\top). \tag{3.12}$$

As for the zap described in Section 3.3, $\sigma_{I,1}$ is deterministically generated from $\sigma_{I,0}$ and at least one of the two CRS's leads to perfectly binding commitments. For simplicity, we will refer to the following matrix of CRS in order to perform Circ proofs:

$$\Sigma \coloneqq [(\sigma_{1,i}, \sigma_{2,j})]_{0 \le i,j \le 1} \coloneqq \begin{bmatrix} (\sigma_{1,0}, \sigma_{2,0}) & (\sigma_{1,0}, \sigma_{2,1}) \\ (\sigma_{1,1}, \sigma_{2,0}) & (\sigma_{1,1}, \sigma_{2,1}) \end{bmatrix}. \tag{3.13}$$

Then, the prover commits to every wire value w_k computing:

$$\mathbf{C}_{I,j,k} \leftarrow \mathsf{Com.C}_I(\sigma_{I,j}, w_k), \tag{3.14}$$

for each $I \in \{1, 2\}$ and each $j \in \{0, 1\}$. Reusing these commitments, the prover now computes four Circ proofs $\pi_{i,j}$ (with $i, j \in \{0, 1\}$)). This boils down to computing, for all $i \in \{0, 1\}$ and all wire indices k:

$$\pi_{i,j,k} \leftarrow \mathsf{Bin.P}\big((\sigma_{1,i}, \sigma_{2,j}),\, (\mathbf{C}_{1,i,k}, \mathbf{C}_{2,j,k}),\, w_k\big), \tag{3.15}$$

and proceeding in the same way for all gates. With a slight abuse of notation, in the explicit construction of Figure 3.8 we denote this whole process with:

$$\pi_{i,j} \leftarrow \mathsf{Circ.P}\big((\sigma_{1,i}, \sigma_{2,j}),\, \phi,\, w\big),$$

keeping in mind that the commitments are not recomputed for each proof, and that instead we are using the commitments $\mathbf{C}_{I,i,k}$ in $\mathbb{G}_I$ to wire w_k under the CRS $\sigma_{I,i}$.
The construction of $\mathsf{ZAP.V}(\phi, \pi)$ is straightforward: Upon receiving a proof

$$(\sigma_{1,0}, \sigma_{2,0},\ (\mathbf{C}_{1,j,k}, \mathbf{C}_{2,j,k})_{i,j\in\{0,1\}},\ [\pi_{i,j}]_{i,j\in\{0,1\}})$$

the verifier computes the correlated CRS's $\sigma_{1,1}, \sigma_{2,1}$ according to Eq. (3.12) and verifies each of the proofs $\pi_{i,j}$ for $i, j \in \{0, 1\}$ using $\mathsf{Circ.V}((\sigma_{1,i}, \sigma_{2,j}),\, \phi,\, \pi_{i,j})$ (using the respective commitments, as we described above). It returns true if all proofs verified.

Theorem 3.4. *Assume SXDH and ADH-KE holds for the asymmetric group generation* GrGen. *Then* ZAP *as defined in Fig. 3.8 is a non-interactive zap that satisfies knowledge soundness and witness indistinguishability.*

Witness indistinguishability of the ZAP proof follows from an hybrid argument analogous to the proof of witness indistinguishability of [GOS06a]. We now prove that the scheme also satisfies computational knowledge soundness.

$\mathsf{ZAP.P}(1^\lambda, \phi, w)$

$\Gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2) \coloneqq \mathsf{GrGen}(1^\lambda)$
for $I = 1, 2$ **do**
  $\sigma_{I,0} \leftarrow \mathsf{Com.G}_I^{(h)}(\Gamma)$
  $\sigma_{I,1} \coloneqq \sigma_{I,0} - ((0,0)^\top, (0, G_I)^\top)$
Compute $(\mathbf{C}_{I,j,k})_{I,j,k}$ as per Eq. (3.14)
for $i, j \in \{0,1\}$ **do**
  $\pi_{i,j} \leftarrow \mathsf{Circ.P}((\sigma_{1,i}, \sigma_{2,j}),\ \phi, w)$
return $\big(\sigma_{1,0}, \sigma_{2,0}, (\mathbf{C}_{I,j,k})_{I,j,k}, [\pi_{i,j}]_{i,j}\big)$

Procedure $\mathsf{Test\text{-}DH}(A, B, C, s)$

$\Gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2) \coloneqq \mathsf{GrGen}(1^\lambda)$
if $sG_1 = A$ **and** $sB = C$
  return 1
if $sG_1 = B$ **and** $sA = C$
  return 1
else
  return 0

Figure 3.8: The non-interactive zap scheme based on SXDH and the procedure for testing DH triples used in the proof.

Proof (computational knowledge soundness). Let $\mathcal{A}$ be the PPT adversary in the game $\mathrm{KSND}_{\mathcal{A},\mathsf{ZAP}}(\lambda)$ able to produce a proof for which ZAP.V returned 1. The proof is of the form:

$$\big(\sigma_{1,0},\ \sigma_{2,0},\ (\mathbf{C}_{1,j,k}, \mathbf{C}_{2,j,k})_{i,j\in\{0,1\}},\ [\pi_{i,j}]_{i,j}\big)$$

where $\pi_{i,j}$ is a valid Circ proof under the CRS $(\sigma_{1,i}, \sigma_{2,j})$ - with $\sigma_{1,1}$ and $\sigma_{2,1}$ derived from $\sigma_{1,0}$ and $\sigma_{2,0}$ as per Equation (3.12).

First, we claim that the extractor is able to find the index $i^* \in \{0,1\}$ of the perfectly binding CRS for $\mathbb{G}_1$. Consider the adversary $\mathcal{A}_{1,0}$ ($\mathcal{A}_{1,1}$, resp.) that behaves as $\mathcal{A}$, but simply outputs the elements (V_0, W_1, W_0) contained in CRS $\sigma_{1,0}$ ($\sigma_{1,1}$, resp.). By ADH-KE there exists an extractor $\mathsf{Ext}_{1,0}$ ($\mathsf{Ext}_{1,1}$, resp.) that outputs a value s_0 (s_1, resp.) in $\mathbb{Z}_p$. If the triple the adversary outputs is a DH triple (which is the case for a perfectly hiding setup), the respective extractor will output the discrete logarithm of one of the first two elements (except with negligible probability). This can be efficiently tested: for a value s output by the extractor, either

$$\begin{aligned} sG_1 &= V_0 \qquad \text{and} \quad sW_1 = W_0\ , \quad \text{or} \\ sG_1 &= W_1 \qquad \text{and} \quad sV_0 = W_0\ , \end{aligned} \tag{3.16}$$

hold. Thus, let $i^* \in \{0,1\}$ be the first value s_{i*} for which Eq. (3.16) does *not* hold. There exists such an i^* because at most one of $\sigma_{1,0}$ and $\sigma_{1,1} = \sigma_{1,0} - ((0,0)^\top, (0, G_1)^\top)$ can be a DH triple and thus a hiding commitment key. In $\mathbb{G}_2$ can can also be at most one hiding commitment key; let j^* be the smallest index such that σ_{2,j^*} is binding. Note that our extractor will not know the value of j^*. The CRS $(\sigma_{1,i^*}, \sigma_{2,j^*})$ is thus of type "perfectly binding".

By soundness of the Bin proof associated to every pair $(\mathbf{C}_{1,i^*,k}, \mathbf{C}_{2,j^*k})$, the committed values $b_{1,k}$ and $b_{2,k}$ satisfy $b_{1,k} = b_{2,k}$ and $b_{1,k}, b_{2,k} \in \{0,1\}$. It now remains to show that these values contained in the commitments corresponding to input wires (which by perfect soundness of Circ constitute a satisfying assignment) can be extracted; in particular, we extract the value from $\mathbf{C}_{1,i^*,k}$ (note that the extractor knows i^*).

Using (once again) the initial adversary $\mathcal{A}$, we can construct multiple adversaries $\mathcal{A}_k^{(b)}$, one for each commitment $\mathbf{C}_{1,i^*,k} = (C_{1,i^*,k,0}, C_{1,i^*,k,1})$ to an input wire, and each possible wire value $b = 0, 1$. The adversary $\mathcal{A}_k^{(b)}$ runs $\mathcal{A}$ and outputs:

$$(V_0,\ C_{1,i^*,k,1},\ C_{1,i^*,k,0}), \qquad (\text{for } \mathcal{A}_k^{(0)})$$
$$(V_0,\ C_{1,i^*,k,1} - W_1,\ C_{1,i^*,k,0} - W_0) \qquad (\text{for } \mathcal{A}_k^{(1)})$$

where $\sigma_{1,i^*} = (V_0, V_1, W_0, W_1)$. Note that if $\mathbf{C}_{1,i^*,k}$ is a commitment to 0 then $(V_0, \mathbf{C}_{1,i^*,k})$ it is of the form (xG_1, rG_1, rxG_1) for some $x, r \in \mathbb{Z}_p$, and thus a DH triple. If $\mathbf{C}_k$ is a commitment to one, then $\mathbf{C} - \mathbf{W}$ is a commitment to 0 and thus $(V_0, \mathbf{C}_{1,i^*,k} - \mathbf{W})$ is a DH triple as above.

By ADH-KE for each adversary $\mathcal{A}_k^{(b)}$ there exists an extractor $\mathsf{Ext}_k^{(b)}$ that outputs some value $s_k^{(b)}$ (with $b = 0, 1$), if $\mathcal{A}_k^{(b)}$ output a DH triple. Using the same reasoning of Eq. (3.16), we can test which of the two triples is a valid DH triple. To do so, we use the procedure Test-DH depicted in Fig. 3.8. For each commitment $\mathbf{C}_{1,i^*,k}$, there exists a single index b_k for which the sub-procedure Test-DH returned 1: if there were more than one we would be violating the perfect binding property of the commitment scheme, if there were none we would be violating the perfect soundness of Bin (as the commitment would be bound to a value different from $0, 1$).

At this point, we are done: the extractor for knowledge soundness runs all above extractors and recovers the bit b_k from every commitment, which is the correct wire value because of soundness of the Circ protocol under a perfectly binding key generation.

As we needed to construct as many extractors as there are input wires in the circuit, the security loss depends on the size of the circuit. □

Chapter 4

Succinct arguments of knowledge: SNARKs

This work was published in the proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. It was completed with co-authors Rosario Gennaro, Michele Minelli, and Anca Nițulescu.

In [GGPR13], Gennaro, Gentry, Parno and Raykova proposed a new, influential characterization of the complexity class NP using *Quadratic Span Programs* (QSPs), a natural extension of span programs defined by Karchmer and Wigderson [KW93]. They show there is a very efficient reduction from boolean circuit satisfiability problems to QSPs. Their work has led to fast progress towards practical verifiable computations. For instance, using Quadratic Arithmetic Programs (QAPs), a generalization of QSPs for arithmetic circuits, Pinocchio [PHGR13] provides evidence that verified remote computation can be faster than local computation. At the same time, their construction is zero-knowledge, enabling the server to keep intermediate and additional values used in the computation private. Optimized versions of SNARK protocols based on QSPs approach are used in various practical applications, including cryptocurrencies such as Zcash [BCG+14], to guarantee anonymity while preventing double-spending.

The QSP approach was generalized in [BCI+13] under the concept of *Linear PCP* (LPCP), a form of interactive ZK proofs where security holds under the assumption that the prover is restricted to compute only linear combinations of its inputs. These proofs can then be turned into (designated-verifier) SNARKs by using an *extractable linear-only* encryption scheme, i.e., an encryption scheme where any adversary can output a valid new ciphertext only if this is an affine combination of some previous encryptions that the adversary had as input (intuitively this "limited malleability" of the encryption scheme, will force the prover into the above restriction).

So far all known zk-SNARKs rely on "classical" pre-quantum assumptions[1]. Yet, there are widely deployed systems relying on zk-SNARKs (for instance, the Zcash cryptocurrency [BCG+14]) which are expected not to withstand cryptanalitic efforts over the course of the next 10 years [ABL+17, Appendix C]. We attempt to make a step forward in this direction by building a designated-verifier zk-SNARK from lattice-based (knowledge) assumptions. Our scheme uses as a main building block encodings that rely on the Learning With Errors (LWE) assumption, initially

[1]We note that the original protocol of Kilian [Kil92] is a zk-SNARK which can be instantiated with a post-quantum assumption since it requires only a collision-resistant hash function – however (even in the best optimized version recently proposed in [BSBHR18]) the protocol does not seem to scale well for even moderately complex computations.

Table 4.1: Security estimates for different choices of LWE parameters (circuit size fixed to $d = 2^{15}$), together with the corresponding sizes of the proof π and of the CRS (when using a seeded PRG for its generation).

security level	λ	n	$\log \alpha$	$\log q$	$\lvert\pi\rvert$	$\lvert\sigma\rvert$	ZK
medium	168	1270	−150	608	0.46 MB	7.13 MB	
	162	1470	−180	736	0.64 MB	8.63 MB	✓
high	244	1400	−150	672	0.56 MB	7.88 MB	
	247	1700	−180	800	0.81 MB	9.37 MB	✓
paranoid	357	1450	−150	800	0.69 MB	9.37 MB	
	347	1900	−180	864	0.98 MB	10.1 MB	✓

proposed by Regev in 2005 [Reg05], and right now the most widespread post-quantum cryptosystem supported by a theoretical proof of security.

SNARGs based on lattices. Recently, in two companion papers [BISW17, BISW18], Boneh et al. provided the first designated-verifier SNARGs construction based on lattice assumptions.

The first paper has two main results: an improvement on the LPCP construction in [BCI$^+$13] and a construction of linear-only encryption based on LWE. The second paper presents a different approach where the information-theoretic LPCP is replaced by a LPCP with multiple provers, which is then compiled into a SNARG again via linear-only encryption. The main advantage of this approach is that it reduces the overhead on the prover, achieving what they call *quasi-optimality*[2]. The stronger notion of knowledge soundness (which leads to SNARKs) can be achieved by replacing the linear-only property with a stronger (extractable) assumption [BCI$^+$13].

Our contributions. In this paper, we frame the construction of Danezis et al. [DFGK14] for Square Span Programs in the framework of "encodings" introduced by Gennaro et al. [GGPR13]. We slightly modify the definition of encoding to accommodate for the noisy nature of LWE schemes. This allows us to have a more fine-grained control over the error growth, while keeping previous example encodings still valid instantiations. Furthermore, SSPs are similar to but simpler than Quadratic Span Programs (QSPs) since they use a single series of polynomials, rather than 2 or 3. We use SSPs to build simpler and more efficient designated-verifier SNARKs and Non-Interactive Zero-Knowledge arguments (NIZKs) for circuit satisfiability (CIRC-SAT).

We think our work is complementary to [BISW17, BISW18]. However, there are several reasons why we believe that our approach is preferable:

Zero-knowledge. The LPCP-based protocols in [BISW17, BISW18] do not investigate the possibility of achieving zero-knowledge. This leaves open the question of whether zk-SNARKs can be effectively instantiated. Considering the LPCP constructed for a QSP satisfiability problem, there is a general transformation to obtain ZK property [BCI$^+$13]. However, in the case of "noisy" encodings, due to possible information leakages in the error term, this transformation cannot be directly applied. Our SNARK construction, being SSP-based, can

[2]This is the first scheme where the prover does not have to compute a cryptographic group operation for each wire of the circuit, which is instead true e.g., in QSP-based protocols.

be made ZK at essentially no cost for either the prover or the verifier. Our transformation is different, exploiting special features of SSPs, and yields a zk-SNARK with almost no overhead. Our construction constitutes the first (designated-verifier) zk-SNARK on lattices.

Weaker assumptions. The linear-only property on encodings introduced in [BCI+13] implies all the security assumptions needed by a SSP-suitable encoding, but the reverse is not known to hold. Our proof of security therefore relies on weaker assumptions and, by doing so, "distills" the minimal known assumptions needed to prove security for SSP, and instantiates them with lattices. We study the relations between our knowledge assumption and the (extractable) linear-only assumption in Section 4.3.

Simplicity and efficiency. While the result in [BISW18] seems asymptotically more efficient than any SSP-based approach, we believe that, for many applications, the simplicity and efficiency of the SSP construction will still provide a concrete advantage in practice. We implemented and tested our scheme: we provide some possible concrete parameters for the instantiation of our zk-SNARKs in Table 4.1, whereas more details on the implementation, along with benchmark results, are presented in Section 4.6.

Technical challenges Although conceptually similar to the original proof of security for QSP-based SNARKs, our construction must incorporate some additional modifications in order to overcome the noise growth of the LWE-based homomorphic operations. These challenges do not arise in the line of work of Boneh et al. [BISW17, BISW18] due to the more general (and stronger) assumption of linear-only encoding (see Section 4.3 for details). Additionally, our construction benefits from the optimizations of SSP-based SNARKs [DFGK14].

Instantiating our encoding scheme with a lattice-based scheme like Regev encryption, differs from [GGPR13] and introduces some technicalities, first in the verification step of the protocol, and secondly in the proof of security. Our encoding scheme is additively homomorphic and allows for linear operations; however, correctness of the encoding is guaranteed only for a limited number of homomorphic operations because of the error growth in lattice-based encoding schemes. More specifically, to compute a linear combination of N encodings, we need to scale some parameters for correctness to hold. Throughout this work we will consider only encodings where a bounded number of homomorphic "linear" operations is allowed, and make sure that this bound is sufficient to perform verification and to guarantee the existence of a security reduction.

Square Span Programs

We characterize NP as Square Span Programs (SSPs) over some field $\mathbb{F}$ of order p. SSPs were introduced first by Danezis et al. [DFGK14].

Definition 4.1 (SSP)**.** *A Square Span Program (SSP) over the field $\mathbb{F}$ is a tuple consisting of $m+1$ polynomials $v_0(x), \ldots, v_m(x) \in \mathbb{F}[x]$ and a target polynomial $t(x)$ such that $\deg(v_i(x)) \leqslant \deg(t(x))$ for all $i = 0, \ldots, m$. We say that the square span program ssp has size m and degree $d = \deg(t(x))$. We say that ssp accepts an input $a_1, \ldots, a_\ell \in \{0,1\}^\lambda$ if and only if there exist $a_{\ell+1}, \ldots, a_m \in \{0,1\}^\lambda$ satisfying:*

$$t(x) \text{ divides } \left(v_0(x) + \sum_{i=1}^{m} a_i v_i(x)\right)^2 - 1.$$

We say that ssp *verifies* a boolean circuit $\mathsf{C} : \{0,1\}^\ell \to \{0,1\}$ if it accepts exactly those inputs $(a_1, \ldots, a_\ell) \in \{0,1\}^\ell$ satisfying $\mathsf{C}(a_1, \ldots, a_\ell) = 1$.

Universal circuit. In the definition, we may see $\mathtt{C}$ as a logical specification of a satisfiability problem. In our zk-SNARK we will split the ℓ inputs into ℓ_u public and ℓ_w private inputs to make it compatible with the universal circuit $C_{\mathcal{U}} : \{0,1\}^{\ell_u} \times \{0,1\}^{\ell_w} \to \{0,1\}$, that take as input an ℓ_u-bit description of a freely chosen circuit $\mathtt{C}$ and an ℓ_w-bit value w, and return 1 if and only if $\mathtt{C}(w) = 1$. Along the lines of [DFGK14], we consider the "public" inputs from the point of view of the prover. For an outsourced computation, they might include both the inputs sent by the clients and the outputs returned by the server performing the computation.

Theorem 4.2 ([DFGK14, Theorem 2])**.** *For any boolean circuit $\mathtt{C}: \{0,1\}^{\ell} \to \{0,1\}$ of m wires and n fan-in 2 gates there exists a degree $d = m+n$ square span program $\mathsf{ssp} = (v_0(x), \ldots, v_m(x), t(x))$ over a field $\mathbb{F}$, of order $p \geq \max(d, 8)$ that verifies $\mathtt{C}$.*

SSP generation. We consider the uniform probabilistic algorithm SSP that, on input a boolean circuit $\mathtt{C} : \{0,1\}^{\ell} \to \{0,1\}$ of m wires and n gates, chooses a field $\mathbb{F}$, with $|\mathbb{F}| \geq \max(d, 8)$ for $d = m + n$, and samples d random elements $r_0, \ldots, r_d \in \mathbb{F}$ to define the target polynomial $t(x) = \prod_{i=0}^{d-1}(x - r_i)$, together with the set of polynomials $\{v_0(x), \ldots, v_m(x)\}$ composing the SSP corresponding to $\mathtt{C}$.

$$(v_0(x), \ldots, v_m(x), t(x)) \leftarrow \mathsf{SSP}(\mathtt{C})$$

Encoding schemes

Encoding schemes for SNARKs were initially introduced in [GGPR13]. Here, we present a variant of this definition that accommodates for encodings with noise.

Definition 4.3 (Encoding Scheme)**.** *An encoding scheme Enc over a field $\mathbb{F}$ is composed of the following algorithms:*

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{E.K}(\Gamma)$*, a key generation algorithm that takes as input the main parameters Γ and outputs some secret state sk together with some public information pk. To ease notation, we are going to assume the message space is always part of the public information and that pk can be derived from sk.*

- $S \leftarrow \mathsf{Enc.E}(\Gamma, param, a)$*, a non-deterministic encoding algorithm mapping a field element a to some encoding space S. Depending on the encoding algorithm, $\mathsf{Enc.E}$ will require either the public information pk generated from $\mathsf{Enc.K}$ or the secret state sk. For our application, it will be the case of sk. To ease notation, we will omit Γ and this additional argument.*

  The output space S is such that $\{[\mathsf{Enc.E}(a)] : a \in \mathbb{F}\}$ partitions S, where $[\mathsf{Enc.E}(a)]$ denotes the set of the possible evaluations of the algorithm $\mathsf{Enc.E}$ on a.

The above algorithms must satisfy the following properties:

d-linearly homomorphic: *there exists a $\mathsf{poly}(\lambda)$ algorithm Eval that, given as input the public parameters pk, a vector of encodings $(\mathsf{Enc.E}(a_1), \ldots, \mathsf{Enc.E}(a_d))$, and coefficients $\mathbf{c} = (c_1, \ldots, c_d) \in \mathbb{F}^d$, outputs a valid encoding of $\mathbf{a} \cdot \mathbf{c}$ with probability overwhelming in λ.*

Quadratic root detection: *there exists an efficient algorithm that, given some parameter δ (either pk or sk), $\mathsf{Enc.E}(a_0), \ldots, \mathsf{Enc.E}(a_t)$, and the quadratic polynomial $\mathsf{pp} \in \mathbb{F}[x_0, \ldots, x_t]$, can distinguish if $\mathsf{pp}(a_1, \ldots, a_t) = 0$. With a slight abuse of notation, we will adopt the writing $\mathsf{pp}(\mathsf{ct}_0, \ldots, \mathsf{ct}_t) = 0$ to denote the quadratic root detection algorithm with inputs δ, $\mathsf{ct}_0, \ldots, \mathsf{ct}_t$, and pp.*

Game q-PKE$_{\mathsf{Enc},Z,\mathcal{A},\mathsf{Ext}_{\mathcal{A}},z}(\lambda)$

$\Gamma \leftarrow \mathsf{Pgen}(1^\lambda)$; $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Enc.K}(\Gamma)$
$(\alpha, s) \leftarrow\$\, \mathbb{F}^*$
$\sigma \leftarrow (\mathsf{pk}, \mathsf{Enc.E}(1), \mathsf{Enc.E}(s), \ldots, \mathsf{Enc.E}(s^q), \mathsf{Enc.E}(\alpha), \mathsf{Enc.E}(\alpha s), \ldots, \mathsf{Enc.E}(\alpha s^q))$
$z \leftarrow Z(\mathsf{pk}, \sigma)$
$(\mathsf{ct}, \hat{\mathsf{ct}}; a_0, \ldots, a_q) \leftarrow (\mathcal{A} \| \mathsf{Ext}_{\mathcal{A}})(\sigma, z)$
return $(\hat{\mathsf{ct}} - \alpha \mathsf{ct} \in [\mathsf{Enc.E}(0)]) \ \wedge \ \mathsf{ct} \notin \left[\mathsf{Enc.E}(\sum_i^q a_i s^i)\right]$

Game q-PKEQ$_{\mathsf{Enc},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)$

$\Gamma \leftarrow \mathsf{Pgen}(1^\lambda)$; $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Enc.K}(\Gamma)$
$s \leftarrow\$\, \mathbb{F}$
$\sigma \leftarrow (\mathsf{pk}, \mathsf{Enc.E}(1), \mathsf{Enc.E}(s), \ldots, \mathsf{Enc.E}(s^q),$
$\quad \mathsf{Enc.E}(s^{q+2}), \ldots, \mathsf{Enc.E}(s^{2q}))$
$(\mathsf{Enc.E}(c), e; b) \leftarrow (\mathcal{A} \| \mathsf{Ext}_{\mathcal{A}})(\sigma)$
if $b = 0$: **return** $e \in [\mathsf{Enc.E}(c)]$
else : **return** $e \notin [\mathsf{Enc.E}(c)]$

Game q-PDH$_{\mathsf{Enc},\mathcal{A}}(\lambda)$

$\Gamma \leftarrow \mathsf{Pgen}(1^\lambda)$; $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Enc.K}(\Gamma)$
$s \leftarrow\$\, \mathbb{F}$
$\sigma \leftarrow (\mathsf{pk}, \mathsf{Enc.E}(1), \mathsf{Enc.E}(s), \ldots, \mathsf{Enc.E}(s^q),$
$\quad \mathsf{Enc.E}(s^{q+2}), \ldots, \mathsf{Enc.E}(s^{2q}))$
$y \leftarrow \mathcal{A}(\sigma)$
return $y \in \left[\mathsf{Enc.E}(s^{q+1})\right]$

Figure 4.1: Games for q-PKE, q-PKEQ, q-PDH assumptions.

Image verification: *there exists an efficiently computable algorithm $\in$ that, given as input some parameter δ (again, either* pk *or* sk*), can distinguish if an element c is a correct encoding of a field element.*

Our specific instantiation of the encoding scheme presents some slight differences with [GGPR13]. In fact, we can allow only for a limited number of homomorphic operations because of the error growth in lattice-based encoding schemes. We note that this modification does not invalidate previous constructions. Sometimes, in order to ease notation, we will employ the writing $\mathsf{ct} \coloneqq \mathsf{Eval}\left(\mathsf{Enc.E}\left(a_i\right)_i, \mathbf{c}\right) = \mathsf{Enc.E}\left(t\right)$, actually meaning that ct is a valid encoding of $t = \sum a_i c_i$; that is, $\mathsf{ct} \in [\mathsf{Enc.E}\left(t\right)]$. It will be clear from the context (and the use of symbol for assignment instead of that for sampling) that the randomized encoding algorithm Enc.E is not actually invoked.

Decoding algorithm. When using a homomorphic encryption scheme in order to instantiate an encoding scheme, we simply define the *decoding algorithm* Enc.D as the decryption procedure of the scheme. More specifically, since we study encoding schemes derived from encryption functions, quadratic root detection and image verification for designated verifiers are trivially obtained by using the decryption procedure Enc.D.

4.1 Cryptographic assumptions

Throughout this work we rely on a number of computational assumptions. All of them are long-standing assumptions in the frame of dlog-hard groups, and have already been generalized in the scope of "encoding schemes" in [GGPR13]. We recall them here for completeness.

The q-power knowledge of exponent assumption (q-PKE) is a generalization of the knowledge of exponent assumption (KEA) introduced by Damgard [Dam92]. It says that given $\mathsf{Enc.E}(s), \ldots,$

$\mathsf{Enc.E}(s^q)$ and $\mathsf{Enc.E}(\alpha s), \ldots, \mathsf{Enc.E}(\alpha s^q)$ for some coefficient α, it is difficult to generate $\mathsf{ct}, \hat{\mathsf{ct}}$ that encode $c, \alpha c$ without knowing the linear combination of the powers of s that produces ct.

Assumption 4.4 (q-PKE)**.** *The q-Power Knowledge of Exponent (q-PKE) assumption holds relative to an encoding scheme* Enc *and for the class $\mathcal{Z}$ of auxiliary input generators if, for every non-uniform PPT auxiliary input generator $Z \in \mathcal{Z}$ and non-uniform* PPT *adversary $\mathcal{A}$, there exists a non-uniform extractor* Ext *such that:*

$$\mathsf{Adv}^{\mathrm{pke}}_{\mathsf{Enc},Z,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) \coloneqq \Pr\left[\text{q-PKE}_{\mathsf{Enc},Z,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) = 1\right] = \mathsf{negl}(\lambda)\,,$$

where $\text{q-PKE}_{\mathsf{Enc},Z,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)$ *is the game depicted in Figure 4.1.*

The q-PDH assumption has been a long-standing, standard q-type assumption [Gro10, BBG05], It basically states that given $(\mathsf{Enc.E}(1), \mathsf{Enc.E}(s), \ldots, \mathsf{Enc.E}(s^q), \mathsf{Enc.E}(s^{q+2}), \ldots, \mathsf{Enc.E}(s^{2q}))$, it is hard to compute an encoding of the missing power $\mathsf{Enc.E}(s^{q+1})$.

Assumption 4.5 (q-PDH)**.** *The q-Power Diffie-Hellman (q-PDH) assumption holds for encoding* Enc *if for all* PPT *adversaries $\mathcal{A}$ we have:*

$$\mathsf{Adv}^{\text{q-PDH}}_{\mathsf{Enc},\mathcal{A}}(\lambda) \coloneqq \Pr\left[\text{q-PDH}_{\mathsf{Enc},\mathcal{A}}(\lambda) = 1\right] = \mathsf{negl}(\lambda)\,,$$

where $\text{q-PDH}_{\mathsf{Enc},\mathcal{A}}(\lambda)$ *is defined as in Figure 4.1.*

Optionally, to achieve strong-soundness (see Remark 2.8), we need an assumption to be able to "compare" adversarially-generated messages. The q-PKEQ assumptions states that for any adversary $\mathcal{A}$ that outputs two ciphertexts, there exists an extractor $\mathsf{Ext}_{\mathcal{A}}$ that can tell whether they encode the same value.

Assumption 4.6 (q-PKEQ)**.** *The q-Power Knowledge of Equality (q-PKEQ) assumption holds for the encoding scheme* Enc *if, for every* PPT *adversary $\mathcal{A}$, there exists an extractor* $\mathsf{Ext}_{\mathcal{A}}$ *such that:*

$$\mathsf{Adv}^{\text{q-PKEQ}}_{\mathsf{Enc},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) \coloneqq \Pr\left[\text{q-PKEQ}_{\mathsf{Enc},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) = 1\right] = \mathsf{negl}(\lambda)\,,$$

where $\text{q-PKEQ}_{\mathsf{Enc},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)$ *is the game depicted in Figure 4.1.*

Te q-PKEQ assumption is needed solely in the case where the attacker has access to a verification oracle. Since the encoding could be non-deterministic, the simulator in the security reduction of Section 4.5 needs to rely on q-PKEQ to simulate the verification oracle. Pragmatically, this assumption allows us to test for equality of two adversarially-produced encodings without having access to the secret key.

Finally, we recall here a well-known assumption for lattices, that we will use to instantiate our quantum-secure encoding scheme.

Assumption 4.7 (dLWE)**.** *The decisional Learning With Errors (dLWE) assumption holds for a parameter generation algorithm* Lgen *if for any* PPT *adversary $\mathcal{A}$:*

$$\mathsf{Adv}^{\mathrm{dLWE}}_{\mathsf{Lgen},\mathcal{A}}(\lambda) \coloneqq \Pr\left[\mathrm{dLWE}_{\mathsf{Lgen},\mathcal{A}}(\lambda) = 1\right] - 1/2 = \mathsf{negl}(\lambda)\,,$$

where $\mathrm{dLWE}_{\mathsf{Lgen},\mathcal{A}}(\lambda)$ *is defined as in Figure 4.2.*

In [Reg05], Regev showed that solving the decisional LWE problem is as hard as solving some lattice problems in the worst case.

Game $\mathrm{dLWE}_{\mathsf{Lgen},\mathcal{A}}(\lambda)$	Oracle ENCODE()
$\Gamma \coloneqq (p, q, n, \alpha) \leftarrow \mathsf{Lgen}(1^\lambda)$	$\mathbf{a} \leftarrow\$ \mathbb{Z}_q^n$
$\mathbf{s} \leftarrow\$ \mathbb{Z}_q^n$	$e \leftarrow \chi_{q\alpha}$
$b \leftarrow\$ \{0, 1\}$	**if** $b = 1$ $c \coloneqq \mathbf{s} \cdot \mathbf{a} + e$
$b' \leftarrow \mathcal{A}^{\text{ENCODE}}(\Gamma)$	**else** $c \leftarrow\$ \mathbb{Z}_q$
return b'	**return** $(\mathbf{a}, c)$

Figure 4.2: The decisional LWE problem for parameters Γ.

4.2 An encoding scheme based on learning with errors

In this section we describe a possible instantiation of the encoding scheme based on learning with errors (LWE).

Lattices. A m-dimensional lattice Λ is a discrete additive subgroup of $\mathbb{R}^m$. For an integer $k < m$ and a rank k matrix $B \in \mathbb{R}^{m \times k}$, $\Lambda(B) = \left\{ B\mathbf{x} \in \mathbb{R}^m \mid \mathbf{x} \in \mathbb{Z}^k \right\}$ is the lattice generated by the columns of B.

Gaussian distribution. For any $\sigma \in \mathbb{R}^+$, let $\rho_\sigma(\mathbf{x}) \coloneqq e^{-\pi \|\mathbf{x}\|^2/\sigma^2}$ be the Gaussian function over $\mathbb{R}^n$ with mean 0 and parameter σ. For any discrete subset $A \subseteq \mathbb{R}^n$ we define $\rho_\sigma(A) \coloneqq \sum_{\mathbf{x} \in A} \rho_\sigma(\mathbf{x})$, the discrete integral of ρ_σ over A. We then define χ_σ, the discrete Gaussian distribution over A with mean 0 and parameter σ as:

$$\chi_\sigma : A \to \mathbb{R}^+ : \mathbf{y} \mapsto \frac{\rho_\sigma(\mathbf{y})}{\rho_\sigma(A)}.$$

We denote by χ_σ^n the discrete Gaussian distribution over $\mathbb{R}^n$ where each entry is independently sampled from χ_σ.

Lattice-based encoding scheme

We propose an encoding scheme Enc that consists of three algorithms as depicted in Figure 4.3. This is a slight variation of the classical LWE cryptosystem initially presented by Regev [Reg05] and later extended in [BV11]. The encoding scheme Enc is described by parameters $\Gamma \coloneqq (q, n, p, \alpha)$, with $q, n, p \in \mathbb{N}$ such that $(p, q) = 1$, and $0 < \alpha < 1$. Our construction is an extension of the one presented in [BV11].

We assume the existence of a PPT algorithm Lgen that, given as input the security parameter in unary 1^λ, outputs an LWE encoding description $\Gamma \coloneqq (p, q, n, \alpha) \leftarrow \mathsf{Lgen}(1^\lambda)$. In real-world encoding, generally parameters are fixed for some well-known values of λ. For the sake of simplicity, we define our encoding scheme with a LWE encoding description Γ and assume that the security parameter λ can be derived from Γ.

Roughly speaking, the public information is constituted by the LWE parameters Γ and an encoding of m is simply an LWE encryption of m. The LWE secret key constitutes the secret state of the encoding scheme.

Basic properties

Correctness. We say that the encoding scheme is (statistically) *correct* if all valid encodings are decoded successfully (with overwhelming probability).

$\mathsf{Enc.K}(\Gamma)$	$\mathsf{Enc.E}(\Gamma, \mathbf{s}, m)$	$\mathsf{Enc.D}(\Gamma, \mathbf{s}, (\mathbf{c}_0, c_1))$
$\mathbf{s} \leftarrow\!\!\$\, \mathbb{Z}_q^n$	$\mathbf{a} \leftarrow\!\!\$\, \mathbb{Z}_q^n$	**return** $(\mathbf{c}_0 \cdot \mathbf{s} + c_1) \pmod p$
return $(\perp,\ \mathbf{s})$	$\sigma \coloneqq q\alpha;\ \ e \leftarrow \chi_\sigma$	
	return $(-\mathbf{a},\ \mathbf{a} \cdot \mathbf{s} + pe + m)$	

Figure 4.3: An encoding scheme based on LWE.

Definition 4.8. *An encoding scheme* Enc *is* correct *if, for any* $\Gamma \in [\mathsf{Lgen}(1^\lambda)]$ *and* $\mathbf{s} \in [\mathsf{Enc.K}(\Gamma)]$ *and* $m \in \mathbb{Z}_p$*:*

$$\Pr[\mathsf{Enc.D}(\Gamma, \mathbf{s}, \mathsf{Enc.E}(\Gamma, \mathbf{s}, m)) \neq m] = \mathsf{negl}(\lambda)\,.$$

We say that an encoding ct *of a message* m *under secret key* $\mathbf{s}$ *is* valid *if* $\mathsf{Enc.D}(\Gamma, \mathbf{s}, \mathsf{ct}) = m$*. We say that an encoding is* fresh *if it is generated through the* $\mathsf{Enc.E}$ *algorithm. We say that an encoding is* stale *if it is not fresh.*

Lemma 4.9 (Correctness)**.** *Let* $\mathsf{ct} = (-\mathbf{a}, \mathbf{a} \cdot \mathbf{s} + pe + m)$ *be an encoding. Then* ct *is a valid encoding of a message* $m \in \mathbb{Z}_p$ *if* $e < \frac{q}{2p}$*.*

Image verification. Using the decryption algorithm $\mathsf{Enc.D}$, and provided with the secret key (i.e., $\delta \coloneqq \mathsf{sk}$), we can implement image verification. The algorithm $\in$ for image verification proceeds as follows: decrypts the encoded element and tests for equality between the two messages.

Quadratic root detection. The algorithm Q for quadratic root detection is straightforward using $\mathsf{Enc.D}$: decrypt the message and evaluate the polynomial, testing if it is equal to 0.

***d*-linearly homomorphicity.** Given a vector of d encodings $\mathsf{ct} \in \mathbb{Z}_q^{d \times (n+1)}$ and a vector of coefficients $\mathbf{c} \in \mathbb{Z}_p^d$, the homomorphic evaluation algorithm is defined as follows: $\mathsf{Eval}(\mathsf{ct}, \mathbf{c}) \coloneqq \mathbf{c} \cdot \mathsf{ct}$.

Technical challenges

Noise growth. During the homomorphic evaluation the noise grows as a result of the operations which are performed on the encodings. Consequently, in order to ensure that the output of Eval is a valid encoding of the expected result, we need to start with a sufficiently small noise in each of the initial encodings.

In order to bound the size of the noise, we first need a basic theorem on the tail bound of discrete Gaussian distributions due to Banaszczyk [Ban95]:

Lemma 4.10 ([Ban95, Lemma 2.4])**.** *For any* $\sigma, T \in \mathbb{R}^+$ *and* $\mathbf{a} \in \mathbb{R}^n$*:*

$$\Pr[\mathbf{x} \leftarrow \chi_\sigma^n\ :\ |\mathbf{x} \cdot \mathbf{a}| \geq T\sigma\,\|\mathbf{a}\|] < 2\exp(-\pi T^2). \tag{4.1}$$

At this point, this corollary follows:

Corollary 4.11. *Let* $\mathbf{s} \leftarrow\!\!\$\, \mathbb{Z}_q^n$ *be a secret key and* $\mathbf{m} = (m_0, \ldots, m_{d-1}) \in \mathbb{Z}_p^d$ *be a vector of messages. Let* ct *be a vector of* d *fresh encodings so that* $\mathsf{ct}_i \leftarrow \mathsf{Enc.E}(\Gamma, \mathbf{s}, m_i)$*, and* $\mathbf{c} \in \mathbb{Z}_p^d$ *be a vector of coefficients. If* $q > 2p^2\sigma\sqrt{\frac{\kappa d}{\pi}}$*, then* $\mathsf{Eval}(\mathbf{c}, \mathsf{ct})$ *outputs a valid encoding of* $\mathbf{m} \cdot \mathbf{c}$ *under the secret key* $\mathbf{s}$ *with probability overwhelming in* κ*.*

Proof. The fact that the message part is $\mathbf{m} \cdot \mathbf{c}$ is trivially true by simple homomorphic linear operations on the encodings. Then the final encoding is valid if the error does not grow too much during these operations. Let $\mathbf{e} \in \mathbb{Z}_p^d$ be the vector of all the error terms in the d encodings, and let $T = \sqrt{\kappa/\pi}$. Then by Lemma 4.10 we have:

$$\Pr\left[\mathbf{e} \leftarrow \chi_\sigma^d \ : \ |\mathbf{e} \cdot \mathbf{c}| \geq \sqrt{\frac{\kappa}{\pi}}\sigma \|\mathbf{c}\|\right] < 2\exp(-\kappa).$$

For correctness we need the absolute value of the final noise to be less than $q/2p$ (cf. Lemma 4.9). Since it holds that $\forall \mathbf{c} \in \mathbb{Z}_p^d$, $\|\mathbf{c}\| \leq p\sqrt{d}$, we can state that correctness holds if:

$$\sqrt{\frac{\kappa}{\pi}}\sigma p\sqrt{d} < \frac{q}{2p}$$

which gives $q > 2p^2\sigma\sqrt{\frac{\kappa d}{\pi}}$. □

Smudging. When computing a linear combination of encodings, the distribution of the error term in the final encoding does not result in a correctly distributed fresh encoding. The resulting error distribution depends on the coefficients used for the linear combination, and despite correctness of the decryption still holds, the error could reveal more than just the plaintext. We combine homomorphic evaluation with a technique called *smudging* [AJL+12], which "smudges out" any difference in the distribution that is due to the coefficients of the linear combination, thus hiding any potential information leak. This technique has been also called "noise flooding" in the past [BPR12].

Lemma 4.12 (Noise Smudging, [Gen09])**.** *Let $B_1 = B_1(\kappa)$ and $B_2 = B_2(\kappa)$ be positive integers. Let $x \in [-B_1, B_1]$ be a fixed integer and $y \leftarrow\!\!\$\ [-B_2, B_2]$. Then the distribution of y is statistically indistinguishable from that of $y + x$, as long as $B_1/B_2 = \mathsf{negl}(\kappa)$.*

Proof. Let Δ denote the statistical distance between the two distributions. By its definition:

$$\Delta = \frac{1}{2}\sum_{v=-(B_1+B_2)}^{B_1+B_2} |\Pr[y = v] - \Pr[y = v - x]| = \frac{1}{2}\left(\sum_{v=-(B_1+B_2)}^{-B_2}\frac{1}{B_2} + \sum_{v=B_2}^{B_1+B_2}\frac{1}{B_2}\right) = \frac{B_1}{B_2}.$$

The result follows immediately. □

In order to preserve the correctness of the encoding scheme while allowing linear evaluations, we need once again q to be large enough to accommodate for the flooding noise. In particular, q will have to be at least superpolynomial in the statistical security parameter κ.

Corollary 4.13. *Let $\mathbf{s} \in \mathbb{Z}_q^n$ be a secret key and $\mathbf{m} = (m_1, \ldots, m_d) \in \mathbb{Z}_p^d$ be a vector of messages. Let ct be a vector of d encodings so that ct_i is a valid encoding of m_i, and $\mathbf{c} \in \mathbb{Z}_p^d$ be a vector of coefficients. Let e_{Eval} be the noise in the encoding output by $\mathsf{Eval}(\mathsf{ct}, \mathbf{c})$ and B_{Eval} a bound on its absolute value. Finally, let $B_{sm} = 2^\kappa B_{\mathsf{Eval}}$, and $e_{sm} \leftarrow\!\!\$\ [-B_{sm}, B_{sm}]$. Then the statistical distance between the distribution of e_{sm} and that of $e_{sm} + e_{\mathsf{Eval}}$ is $2^{-\kappa}$. Moreover, if $q > 2p\,B_{\mathsf{Eval}}(2^\kappa + 1)$ then the result of $\mathsf{Eval}(\mathsf{ct}, \mathbf{c}) + (\mathbf{0}, e_{sm})$ is a valid encoding of $\mathbf{m} \cdot \mathbf{c}$ under the secret key $\mathbf{s}$.*

Proof. The claim on the statistical distance follows immediately from Lemma 4.12 and the fact that the message part is $\mathbf{m} \cdot \mathbf{c}$ is true by simple homomorphic linear operations on the encodings. In order to ensure that the final result is a valid encoding, we need to make sure that the error in

Procedure $\mathsf{test\text{-}error}(\Gamma, \mathbf{s}, (\mathbf{c}_0, c_1))$

$e' \coloneqq (\mathbf{c}_0 \cdot \mathbf{s} + c_1) \;//\; p$

return $(Eq.\ (4.3))$

Figure 4.4: The error testing procedure.

this output encoding remains smaller than $q/2p$. The final error is upper bounded by $B_{\mathsf{Eval}} + B_{sm}$, so we have:

$$B_{\mathsf{Eval}} + B_{sm} < \frac{q}{2p} \implies B_{\mathsf{Eval}} + 2^{\kappa} B_{\mathsf{Eval}} < \frac{q}{2p} \implies q > 2p\, B_{\mathsf{Eval}}\,(2^{\kappa} + 1)\,.$$

□

Error testing. By making non-blackbox use of our LWE encoding scheme, it is possible to define an implementation of the function $\mathsf{test\text{-}error}$ in order to guarantee the existence of a security reduction from adversarially-generated proofs. In fact, it is not sufficient to show that a series of homomorphic operations over a forged proof can break one of the assumptions. We must also guarantee that these manipulations do not alter the correctness of the encoded value. In the specific case of LWE encodings, it is sufficient to use the secret key, recover the error, and enforce an upper bound on its norm. A possible implementation of $\mathsf{test\text{-}error}$ is displayed in Figure 4.4.

Other requirements for security reduction. The following lemma will be needed later during the security proof. It essentially defines the conditions under which we can take an encoding, add a smudging term to its noise, sum it with the output of an execution of Eval and finally multiply the result by an element in $\mathbb{Z}_p$.

Lemma 4.14 (For reduction)**.** *Let* $\mathbf{s}$, ct, $\mathbf{c}$, e_{Eval}, B_{Eval} *be as in Corollary 4.13, and let* $\mathsf{ct}' = (-\mathbf{a}', \mathbf{s} \cdot \mathbf{a}' + pe' + m')$ *be a valid encoding of a message* $m' \in \mathbb{Z}_p$ *with noise* e' *bounded by* B_e*. Let* $B_{sm} = 2^{\kappa} B_e$ *and* $e_{sm} \leftarrow\!\!\$\ [-B_{sm}, B_{sm}]$ *be a "smudging noise". Then, if* $q > 2p^2\,((2^{\kappa}+1)\,B_e + B_{\mathsf{Eval}})$*, it is possible to add the smudging term* e_{sm} *to* ct'*, sum the result with the output of* $\mathsf{Eval}\,(\mathsf{ct}, \mathbf{c}\,)$*, multiply the outcome by a coefficient bounded by* p*, and obtain a valid encoding of* $k\,(\mathbf{m} \cdot \mathbf{c} + m')$*.*

Proof. The correctness of the message part comes immediately from performing homomorphic linear operations on encodings, and the final output is valid if the noise remains below a certain threshold. After adding the smudging term and performing the sum, the noise term is at most $B_e + B_{sm} + B_{\mathsf{Eval}} = (2^{\kappa}+1)\,B_e + B_{\mathsf{Eval}}$. After the multiplication by a coefficient bounded by p, it is at most $p\,((2^{\kappa}+1)\,B_e + B_{\mathsf{Eval}})$. Thus, the encoding is valid if:

$$p\,((2^{\kappa}+1)\,B_e + B_{\mathsf{Eval}}) < \frac{q}{2p}, \tag{4.2}$$

which immediately gives the result. □

Conditions on the modulus q**.** Corollaries 4.11 and 4.13 and Lemma 4.14 give the conditions that the modulus q has to respect in order to allow for all the necessary computations. In particular, Corollary 4.11 gives the condition to be able to homomorphically evaluate a linear combination of fresh encodings through the algorithm Eval; Corollary 4.13 gives the condition to be able to add a smudging noise to the result of such an evaluation; Lemma 4.14 gives a condition that will have

to be satisfied in the security reduction. They are ordered from the least stringent to the most stringent, so the condition that must be satisfied in the end is the one given by Lemma 4.14:

$$q > 2p^2 \left((2^\kappa + 1) B_e + B_{\mathsf{Eval}} \right) \tag{4.3}$$

Leftover hash lemma (LHL). We now recall the definition of min-entropy, and the famous "leftover hash lemma" introduced by Impagliazzo et al. [HILL99].

Definition 4.15 (Min-entropy)**.** *The min-entropy of a random variable X is defined as:*

$$H_\infty(X) = -\log\left(\max_x \Pr[X = x]\right)$$

Lemma 4.16 (Leftover hash lemma)**.** *Assume a family of functions $\left\{\mathcal{H}_x : \{0,1\}^n \to \{0,1\}^\ell\right\}_{x \in X}$ is* universal*, i.e., $\forall a \neq b \in \{0,1\}^n$:*

$$\Pr_{x \in X}[\mathcal{H}_x(a) = \mathcal{H}_x(b)] = 2^{-\ell}.$$

Then, for any random variable Y:

$$\Delta\left((X, \mathcal{H}_X(Y)), (X, U_\ell)\right) \leq \frac{1}{2}\sqrt{2^{-H_\infty(Y)} \cdot 2^\ell},$$

where $U_\ell \leftarrow\!\!\$\, \{0,1\}^\ell$.

Zero Knowledge. We now present a version of the LHL that will be useful later in this work, when proving the zero knowledge property of our construction. In a nutshell, it says that a random linear combination of the columns of a matrix is statistically close to a uniformly random vector, for some particular choice of coefficients.

Lemma 4.17 ("Specialized" leftover hash lemma)**.** *Let n, p, q, d be non-negative integers. Let $\mathbf{A} \leftarrow\!\!\$\, \mathbb{Z}_q^{n \times d}$, and $\mathbf{r} \leftarrow\!\!\$\, \mathbb{Z}_p^d$. Then we have:*

$$\Delta\left((\mathbf{A}, \mathbf{Ar}), (\mathbf{A}, \mathbf{u})\right) \leq \frac{1}{2}\sqrt{p^{-d} \cdot q^n},$$

where $\mathbf{Ar}$ is computed modulo q, and $\mathbf{u} \leftarrow\!\!\$\, \mathbb{Z}_q^n$.

Proof. For the vector $\mathbf{r}$, we have that $H_\infty(\mathbf{r}) = d \log p$. Then the proof is immediate from Lemma 4.16:

$$\Delta\left((\mathbf{A}, \mathbf{Ar}), (\mathbf{A}, \mathbf{u})\right) \leq \frac{1}{2}\sqrt{2^{-d\log p} \cdot q^n} = \frac{1}{2}\sqrt{p^{-d} \cdot q^n}.$$

□

4.3 Lattices and assumptions

In this section, we analyze the assumptions that we make in this work and how they relate to the assumptions made in previous works. At a first glimpse, it might seem unjustified to have brought assumptions often used in the dLog setting into the lattice domain, where they are highly non-standard. Despite this fact, in this section we argue (i) that the q-PKE and q-PDH assumptions are weaker than the targeted linear-only malleability of [BCI+13, BISW17], and (ii) which consequences an attack on those assumptions would have.

Over the course of the last years, a long line of research in lattice-based cryptography has been trying to develop fully-homomorphic encryption schemes and bilinear pairing maps. So far,

Game $\text{EXT-LO}_{\mathsf{Enc},\mathsf{M},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)$

$(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Enc.K}(1^\lambda, 1^d)$

$(m_1,\ldots,m_d) \leftarrow \mathsf{M}(1^\lambda, 1^d)$

$\sigma \leftarrow (\mathsf{Enc.E}(m_1),\ldots,\mathsf{Enc.E}(m_d))$

$(\mathsf{ct}; a_0,\ldots,a_d) \leftarrow (\mathcal{A}\|\mathsf{Ext}_{\mathcal{A}})(\sigma)$

return $\mathsf{ct} \notin \left[\mathsf{Enc.E}(a_0 + \sum_{i=1}^{d} a_i m_i)\right]$

Figure 4.5: Game for Extractable Linear-Only target malleability.

no bilinear map is known in the context of lattices, and some have argued that its existence would lead to efficient cryptographic primitives such as multilinear maps and indistinguishability obfuscation (iO). Furthermore, although there exist FHE schemes based on lattices, it is not clear how to achieve it without giving away additional information such as encryption of the secret key itself. Showing that it is possible to indeed compute non-linear homomorphisms on top of Regev's encryption scheme would be a major breakthrough in both research areas. We see this as a win-win situation.

Moreover, our assumptions are *weaker* than previously employed assumptions for lattice-based SNARGs. Indeed, the linear-only assumption of [BCI+13, BISW17] informally states that an adversary can only perform affine operations over the encodings provided as input. More specifically:

Definition 4.18 (EXT-LO, [BCI+13])**.** *An encoding scheme* Enc *satisfies* extractable linear-only target malleability *if for all* PPT *adversaries* $\mathcal{A}$ *and plaintext generation algorithm* M *there exists an efficient extractor* Ext *such that the advantage:*

$$\mathsf{Adv}^{\text{ext-lo}}_{\mathsf{Enc},\mathsf{M},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) \coloneqq \Pr\left[\text{EXT-LO}_{\mathsf{Enc},\mathsf{M},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) = 1\right] = \mathsf{negl}(\lambda)\,,$$

where $\text{EXT-LO}_{\mathsf{Enc},\mathsf{M},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)$ *is defined as in Figure 4.5.*

We note that, despite [BCI+13] presents the above assumption for so-called *linear-only encryption schemes*, all such schemes are also encodings satisfying the properties of Definition 4.3.

It is not immediately clear to see what does this assumption imply in the case of LWE encodings (like the one we presented in section Section 4.2) or the one in [LP11], used in [BISW17]. Consider for example a set of parameters Γ allowing for $d-1$ homomorphic operations modulo p and the adversary $\mathcal{A}$ that, upon receiving as input d ciphertexts, computes d homomorphic linear operations on them. With non-negligible probability the error wraps around the modular representation, leading to a "decryptable" encoding (any element of $\mathbb{Z}_q^{n+1}$ is a valid encoding) but for which the adversary does not know an affine map. The authors of [BISW17] suggest to use *double-encryption* in these cases, i.e., present two different encodings of each value, and ask the adversary to homomorphically evaluate these terms twice. If the two ciphertexts do not encode the same element, the game is lost. Obviously, this comes at the cost of doubling the size of each encoding and doubling the computation time for the prover and the verifier.

Theorem 4.19. *If* Enc *is an* IND-CPA *extractable linear-only encoding scheme, it satisfies* q-PDH*.*

Proof. Let us consider an adversary $\mathcal{A}^{\text{PDH}}$ for the q-PDH assumption. We show that there exists an adversary able to break IND-CPA.

Consider the PPT machine $\mathcal{A}$ that samples uniformly at random two field elements, s_0 and s_1, then submits the two distinct chosen plaintexts s_0^{q+1}, s_1^{q+1} to the IND-CPA challenger. $\mathcal{A}$ queries

the IND-CPA encoding oracle with s_0^k, s_1^k for $k = 0, \ldots, q, q+2, \ldots, 2q$. The oracle gives back some encodings $\sigma := \left(\mathsf{Enc.E}(1), \mathsf{Enc.E}(s_b), \ldots, \mathsf{Enc.E}(s_b^q), \mathsf{Enc.E}(s_b^{q+2}), \ldots, \mathsf{Enc.E}(s_b^{2q})\right)$ for $b \in \{0,1\}$. $\mathcal{A}$ runs the q-PDH adversary on σ, thus obtaining (with non-negligible probability) some encoding $\mathsf{ct} \in \left[\mathsf{Enc.E}(s_b^{q+1})\right]$. By EXT-LO, there exists an extractor $\mathsf{Ext}^{\mathsf{LO}}$ which, given as input σ and the same random coins of the adversary $\mathcal{A}^{\text{PDH}}$, returns a polynomial p such that $p(s_b) = s_b^{q+1}$. Let $f(x) := p(x) - x^{q+1}$. By q-PDH, $f(s_b) = 0$; by Schwartz-Zippel lemma, $f(s_{1-b}) \neq 0$ with probability $1 - 2q/|\mathbb{F}| = 1 - \mathsf{negl}(\lambda)$. $\mathcal{A}$ returns the bit b^* such that $f(s_{b^*}) = 0$, thus solving the IND-CPA challenge with overwhelming probability. □

Theorem 4.20. *If* Enc *is an* IND-CPA *extractable linear-only encoding scheme, it satisfies* q-PKE.

Proof. We will show that Enc satisfies q-PKE, meaning there is no $\mathcal{A}^{\text{PKE}} \| \mathsf{Ext}_{\mathcal{A}}$ able to win the q-PKE game (cf. Figure 4.1).

Suppose by contradiction that there exists an adversary $\mathcal{A}^{\text{PKE}}$ able to produce a valid output $\mathsf{ct}, \hat{\mathsf{ct}}$, i.e., such that $\alpha\mathsf{ct} - \hat{\mathsf{ct}} = 0$. We show that as a consequence there exists an extractor $\mathsf{Ext}_{\mathcal{A}}$ that outputs the correct linear combination with non negligible probability.

Let M be the plaintext generation algorithm that, upon receiving the computational security parameter λ and $d = 2q + 2$ in unary form, samples $s \leftarrow\!\$\, \mathbb{F}$ and outputs plaintexts $1, s, \ldots, s^q$. Let $\sigma \leftarrow (\mathsf{Enc.E}(1), \mathsf{Enc.E}(s), \ldots, \mathsf{Enc.E}(s^q), \mathsf{Enc.E}(\alpha), \mathsf{Enc.E}(\alpha s), \ldots, \mathsf{Enc.E}(\alpha s^q))$. The adversary $\mathcal{A}^{\text{PKE}}$, when run on this input σ, outputs (with non-negligible probability) $\mathsf{ct}, \hat{\mathsf{ct}}$ such that $\alpha\mathsf{ct} - \hat{\mathsf{ct}} = 0$ (via quadratic root detection algorithm).

Let us define the adversaries $\mathcal{B}_0$ and $\mathcal{B}_1$ for the game EXT-LO that, upon receiving as input σ, run the same instantiation of $\mathcal{A}^{\text{PKE}}$ and output ct - respectively $\hat{\mathsf{ct}}$. By our claim of linear-only property, there exist the extractors Ext_0 and Ext_1 for $\mathcal{B}_0$ and $\mathcal{B}_1$, respectively, outputting $a_0, \ldots, a_q, b_0, \ldots, b_q$ and $a'_0, \ldots, a'_q, b'_0, \ldots, b'_q$ such that:

$$\mathsf{ct} \in \left[\mathsf{Enc.E}\left(\textstyle\sum_i^d a_i s^i + \sum_i^d b_i \alpha s^i\right)\right], \quad \hat{\mathsf{ct}} \in \left[\mathsf{Enc.E}\left(\textstyle\sum_i^d a'_i s^i + \sum_i^d b'_i \alpha s^i\right)\right]$$

with non negligible probability.

Knowing that $\alpha\mathsf{ct} - \hat{\mathsf{ct}} = 0$ implies either that the polynomial:

$$P(X,Y) = X^2 \textstyle\sum_i^d b_i Y^i + X \sum_i^d (a_i - b'_i) Y^i - \sum_i^d a'_i Y^i$$

is the zero polynomial, or that (α, s) are roots of $P(X,Y)$. The second case is ruled out by semantic security of the encoding scheme and Schwartz-Zippel lemma, by a reasoning similar to the proof of Theorem 4.19.

The case where $P(X,Y) = 0$ gives us $b_i = a'_i = 0, a_i = b'_i \forall i = 0, \ldots, q$. Therefore, we are able to define an extractor $\mathsf{Ext}_{\mathcal{A}}$ for q-PKE that outputs the coefficients a_i of the linear combination with non-negligible probability, showing that any successful adversary against q-PKE able to output $\mathsf{ct}, \hat{\mathsf{ct}}$ such that $\alpha\mathsf{ct} - \hat{\mathsf{ct}} \in [\mathsf{Enc.E}(0)]$, has knowledge of the coefficients a_i such that $\mathsf{ct} \in \left[\mathsf{Enc.E}\left(\sum_i^d a_i s^i\right)\right]$.

□

4.4 Our designated-verifier zk-SNARK

Let Enc be an encoding scheme (Definition 4.3). Let $\mathtt{C}$ be some circuit taking as input an ℓ_ϕ-bit string and outputting 0 or 1. Let $\ell := \ell_\phi + \ell_w$, where ℓ_ϕ is the length of the "public" input, and ℓ_w the length of the private input. The value m corresponds to the number of wires in $\mathtt{C}$ and n to the number of fan-in 2 gates. Let $d := m + n$. We construct a zk-SNARK scheme for any relation $\mathsf{R}_{\mathtt{C}}$ on pairs $(\phi, w) \in \{0,1\}^{\ell_\phi} \times \{0,1\}^{\ell_w}$ that can be computed by a polynomial size circuit $\mathtt{C}$ with m wires and n gates. Our protocol is formally depicted in Figure 4.6.

Setup $\Pi.\mathsf{G}(\Gamma)$

$\alpha, \beta, s \leftarrow\$ \mathbb{F}$; $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Enc.K}(\Gamma)$
$(v_0, \ldots, v_m(x), t(x)) \leftarrow \mathsf{SSP}(\mathtt{C})$
Compute σ as per Eq. (4.4)
$\mathsf{vrs} \coloneqq (\Gamma, \mathsf{sk})$; $\tau \coloneqq (\Gamma, \mathsf{sk}, s, \alpha, \beta)$
return $(\sigma, \mathsf{vrs}, \tau)$

Verifier $\Pi.\mathsf{V}(\mathsf{vrs}, u, \pi)$

$(H, \hat{H}, \hat{V}, V_w, B_w) \coloneqq \pi$
$(a_1, a_2, \ldots a_{\ell_\phi}) \coloneqq u$; $(\Gamma, \mathsf{sk}) \coloneqq \mathsf{vrs}$
$w_s \coloneqq \mathsf{Enc.D}(V_w)$; $b_s \coloneqq \mathsf{Enc.D}(B_w)$
$h_s \coloneqq \mathsf{Enc.D}(H)$; $\hat{h}_s \coloneqq \mathsf{Enc.D}(\hat{H})$
$\hat{v}_s \coloneqq \mathsf{Enc.D}(\hat{V})$; $t_s \coloneqq t(s)$
$v_s \coloneqq v_0(s) + \sum_{i=1}^{\ell_\phi} a_i v_i(s) + w_s$
Check Eqs. (eq-pke) to (eq-lin)
return $\mathsf{test\text{-}error}(\Gamma, \mathsf{sk}, B_w)$

Prover $\Pi.\mathsf{P}(\sigma, \phi, w)$

$(\Gamma, \mathsf{ssp}, \mathsf{pk}, \ldots) \coloneqq \sigma$
$(v_0, \ldots, v_m(x), t(x)) \coloneqq \mathsf{ssp}$
$u \coloneqq (a_1, \ldots, a_{\ell_\phi}) \in \{0,1\}^{\ell_\phi}$;
$w \coloneqq (a_{\ell_\phi+1}, \ldots, a_m)$
$\nu(x) \coloneqq v_0(x) + \sum_{i=1}^{m} a_i v_i(x) + \gamma t(x)$
$v_{\mathsf{mid}}(x) \coloneqq \sum_{i>\ell_\phi}^{m} a_i v_i(x) + \gamma t(x)$
$h(x) = (\nu(x)^2 - 1)/t(x)$
// Compute the proof terms as per Eq. (4.6)
$H \coloneqq \mathsf{Eval}((\mathsf{Enc.E}(s^i))_i^d,\ (h_i)_i^d) = \mathsf{Enc.E}(h(s))$
$\hat{H} \coloneqq \mathsf{Eval}((\mathsf{Enc.E}(\alpha s^i))_i^d,\ (h_i)_i^d) = \mathsf{Enc.E}(\alpha h(s))$
$\hat{V} \coloneqq \mathsf{Eval}((\mathsf{Enc.E}(\alpha s^i)_i^d,\ (\nu_i)_i^d) = \mathsf{Enc.E}(\alpha \nu(s))$
$B_w \coloneqq \mathsf{Eval}((\mathsf{Enc.E}(\beta v_i(s)))_i^m \| (\mathsf{Enc.E}(\beta t(s))),\ (a_i)_i^m \| (\gamma))$
$V_w \coloneqq \mathsf{Eval}((\mathsf{Enc.E}(s^i))_i^d,\ (v_{\mathsf{mid}\,i})_i^d) = \mathsf{Enc.E}(v_{\mathsf{mid}}(s))$
Apply smudging on $H, \hat{H}, \hat{V}, B_w, V_w$
return $(H, \hat{H}, \hat{V}, V_w, B_w)$

Figure 4.6: Our zk-SNARK protocol Π.

CRS generation. The setup algorithm G takes as input some complexity 1^λ in unary form and the circuit $\mathtt{C} : \{0,1\}^{\ell_\phi} \times \{0,1\}^{\ell_w} \to \{0,1\}$. It generates a square span program of degree $d = m+n$ over a field $\mathbb{F}$, of size $|\mathbb{F}| \geq d$ that verifies $\mathtt{C}$ by running:

$$\mathsf{ssp} \coloneqq (v_0(x), \ldots, v_m(x), t(x)) \leftarrow \mathsf{SSP}(\mathtt{C})$$

Then, it runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Enc.K}(1^\lambda)$ using the encoding scheme Enc. Finally, it samples $\alpha, \beta, s \leftarrow \mathbb{F}$ such that $t(s) \neq 0$, and returns the CRS:

$$\begin{aligned} \sigma \coloneqq \Big(\Gamma,\ \mathsf{ssp},\ \mathsf{pk},\ & \mathsf{Enc.E}(1), \mathsf{Enc.E}(s), \ldots, \mathsf{Enc.E}(s^d), \\ & \mathsf{Enc.E}(\alpha), \mathsf{Enc.E}(\alpha s), \ldots, \mathsf{Enc.E}(\alpha s^d), \\ & \mathsf{Enc.E}(\beta t(s)), (\mathsf{Enc.E}(\beta v_i(s)))_{i=\ell_\phi+1}^m \Big) \end{aligned} \tag{4.4}$$

The error for each of these encodings has to be chosen carefully. In a nutshell, we need to intentionally increase the magnitude of the noise in some encodings, in order to mimic its distribution in the simulated CRS provided to the adversary in the security reduction. Failing to do so results in limiting the adversary's ability to perform homomorphic operations on the CRS and, thus, in a flawed proof. We defer further analysis on this point to Section 4.6. The verification string vrs consists of the secret key sk, and the CRS σ. The trapdoor τ consists of the secret key sk and the secrets s, α, β.

Prover. The prover algorithm, on input some statement $\phi \coloneqq (a_1, \ldots, a_{\ell_\phi})$, computes a witness $w \coloneqq (a_{\ell_\phi+1}, \ldots, a_m)$ such that $(\phi \| w) = (a_1, \ldots, a_m)$ is a satisfying assignment for the circuit $\mathtt{C}$. The $(a_i)_i$ are such that:

$$t(x) \text{ divides } \left(v_0(x) + \sum_{i=1}^{m} a_i v_i(x) \right)^2 - 1,$$

as per Theorem 4.2. Then, it samples $\gamma \leftarrow\!\!\$\, \mathbb{F}$ and sets $\nu(x) \coloneqq v_0(x) + \sum_{i=1}^{m} a_i v_i(x) + \gamma t(x)$. Let:

$$h(x) \coloneqq \frac{(v_0(x) + \sum_i^m a_i v_i(x) + \gamma t(x))^2 - 1}{t(x)} = \frac{\nu(x)^2 - 1}{t(x)}, \tag{4.5}$$

whose coefficients can be computed from the polynomials provided in the ssp; them by linear evaluation it is possible to obtain:

$$\begin{aligned} H &\coloneqq \mathsf{Enc.E}(h(s)), \qquad \hat{H} \coloneqq \mathsf{Enc.E}(\alpha h(s)), \qquad \hat{V} \coloneqq \mathsf{Enc.E}\left(\alpha \nu(s)\right), \\ V_w &\coloneqq \mathsf{Enc.E}\left(\sum_{i=\ell_\phi+1}^{m} a_i v_i(s) + \gamma t(s)\right), \\ B_w &\coloneqq \mathsf{Enc.E}\left(\beta\left(\sum_{i=\ell_\phi+1}^{m} a_i v_i(s) + \gamma t(s)\right)\right). \end{aligned} \tag{4.6}$$

In fact, the encoding H - respectively, $\hat{H}$ - can be computed from the encodings of $1, s, \ldots, s^d$ - respectively, $\alpha, \alpha s, \ldots, \alpha s^d$ - and the coefficients of Equation (4.5). The element $\hat{V}$ can be computed from the encodings of $\alpha s, \ldots, \alpha s^d$. Finally, V_w - respectively, B_w - can be computed from the encodings of $s, \ldots, s^d$ - respectively, $\beta t(s), \beta v_{\ell_\phi+1}(s), \ldots, \beta v_m(s)$. All these linear evaluations involve at most $d+1$ terms and the coefficients are bounded by p. Using the above elements, the prover returns a proof $\pi \coloneqq (H, \hat{H}, \hat{V}, V_w, B_w)$.

Verifier. Upon receiving a proof π and a statement $\phi = (a_1, \ldots, a_{\ell_\phi})$, the verifier, in possession of the verification key vrs (that implicitly contains the σ), proceeds with the following verifications. First, it uses the quadratic root detection algorithm of the encoding scheme Enc to verify that the proof satisfies:

$$\hat{h}_s - \alpha h_s = 0 \text{ and } \hat{v}_s - \alpha v_s = 0, \tag{eq-pke}$$

$$(v_s^2 - 1) - h_s t_s = 0, \tag{eq-div}$$

$$b_s - \beta w_s = 0. \tag{eq-lin}$$

where $(h_s, \hat{h}_s, \hat{v}_s, w_s, b_s)$ are the values encoded in $(H, \hat{H}, \hat{V}, V_w, B_w) \coloneqq \pi$ and t_s, v_s are computed as $t_s \coloneqq t(s)$ and $v_s \coloneqq v_0 + \sum_{i=1}^{\ell_\phi} a_i v_i(s) + w_s$.

Then, the verifier checks whether it is still possible to perform some homomorphic operations, using the test-error procedure, implemented in Figure 4.4 for the specific case of lattice encodings. More precisely, the verifier tests whether it is still possible to add another encoding and multiply the result by an element bounded by p, without compromising the correctness of the encoded element. This will guarantee the existence of a reduction in the knowledge soundness proof of Section 4.5. If all above checks hold, the verifier returns 1. Otherwise, return 0.

Remark 4.21. Instantiating our encoding scheme on top of a "noisy" encryption scheme like Regev's introduces multiple technicalities that affect the protocol, the security proof, and the parameters' choice. For instance, in order to compute a linear combination of d encodings via Eval we need to scale down the error parameter and consequently increase the parameters q and n in order to maintain correctness and security. Similarly, the distributions of the error terms and the random vectors are affected by the homomorphic evaluation, and we must guarantee that the resulting terms are still simulatable. All these issues will be formally addressed in Section 4.5, and then analyzed more pragmatically in Section 4.6.

4.5 Proofs of security

In this section, we prove our main theorem:

Theorem 4.22. *If the* q-PKE, q-PKEQ *and* q-PDH *assumptions hold for the encoding scheme* Enc*, the protocol* Π *on* Enc *is a zk-SNARK with statistical completeness, statistical zero knowledge and computational knowledge soundness.*

Statistical completeness. Corollary 4.11 states the conditions on Γ for which the homomorphically computed encodings are valid with probability at least $1 - \mathsf{negl}(\kappa)$. Lemma 4.14 affirms that correctly generated proofs satisfy Equation (4.2) with probability overwhelming in κ. Therefore $\mathsf{test\text{-}error}$ returns true and completeness follows trivially by Theorem 4.2. □

Knowledge soundness

Proof of computational knowledge soundness. Let $\mathcal{A}^{\Pi}$ be the PPT adversary in the game for knowledge soundness (Figure 2.2) able to produce a proof π for which $\Pi.\mathsf{V}$ returned 1. We first claim that it is possible to extract the coefficients of the polynomial $v(x)$ corresponding to the values v_s encoded in V. The setup algorithm first generates the parameters $(\mathsf{pk}, \mathsf{sk})$ of an encoding scheme Enc and picks $\alpha, \beta, s \in \mathbb{F}$, which are used to compute $\mathsf{Enc.E}(1), \mathsf{Enc.E}(s), \ldots, \mathsf{Enc.E}(s^d), \mathsf{Enc.E}(\alpha)$, $\mathsf{Enc.E}(\alpha s), \ldots, \mathsf{Enc.E}(\alpha s^d)$. Fix some circuit C, and let ssp be an SSP for C. Let $\mathcal{A}^{\mathrm{PKE}}$ be the d-PKE adversary, that takes as input a set of encodings:

$$\sigma := \left(\mathsf{pk}, \mathsf{Enc.E}(1), \mathsf{Enc.E}(s), \ldots, \mathsf{Enc.E}(s^d), \mathsf{Enc.E}(\alpha), \mathsf{Enc.E}(\alpha s), \ldots, \mathsf{Enc.E}(\alpha s^d)\right).$$

The auxiliary input generator Z is the PPT machine that upon receiving as input σ, samples $\beta \leftarrow\!\!\$\, \mathbb{Z}_p$, constructs the remaining terms of the CRS (as per Equation (4.4)), and outputs them in z using ssp. Thus, $\mathcal{A}^{\mathrm{PKE}}$ sets $\sigma := (\mathsf{ssp} \| \sigma \| z)$ and invokes $\mathcal{A}^{\Pi}(\sigma)$. As a result, it obtains a proof $\pi = (H, \hat{H}, \hat{V}, V_w, B_w)$. On this proof, it computes:

$$V := \mathsf{Enc.E}\left(v_0 + \sum_{i=1}^{\ell_\phi} a_i v_i(s) + w_s\right) = V_w + v_0 + \sum_{i=1}^{\ell_\phi} a_i v_i(s). \tag{4.7}$$

where w_s is the element encoded in V_w. Finally, $\mathcal{A}^{\mathrm{PKE}}$ returns $(\hat{V}, V)$. If the adversary $\mathcal{A}$ outputs a valid proof, then by verification equation Eq. (eq-pke) it holds that the two encodings $(V, \hat{V})$ encode values $v_s, \hat{v}_s$ such that $\hat{v}_s - \alpha v_s = 0$. Therefore, by q-PKE assumption there exists an extractor $\mathsf{Ext}^{\mathrm{PKE}}$ that, using the same input (and random coins) of $\mathcal{A}^{\mathrm{PKE}}$, outputs a vector $(c_0, \ldots, c_d) \in \mathbb{F}^{d+1}$ such that V is an encoding of $\sum_{i=0}^{d} c_i s^i$ and $\hat{V}$ is an encoding of $\sum_{i=0}^{d} \alpha c_i s^i$. In the same way, it is possible to recover the coefficients of the polynomial $h(x)$ used to construct $(H, \hat{H})$, the first two elements of the proof of $\mathcal{A}^{\Pi}$ (again, by Eq. (eq-pke)).

Our witness extractor Ext^{Π}, given σ, emulates the extractor $\mathsf{Ext}^{\mathrm{PKE}}$ above on the same input σ, using as auxiliary information z the rest of the CRS given as input to Ext^{Π}. By the reasoning discussed above, Ext^{Π} can recover $(c_0, \ldots, c_d)$ coefficients extracted from the encodings $(V, \hat{V})$. Consider now the polynomial $v(x) := \sum_{i=0}^{d} c_i x^i$. If it is possible to write the polynomial as $v(x) = v_0(x) + \sum_i^m a_i v_i(x) + \delta t(x)$ such that $(a_1, \ldots, a_m) \in \{0,1\}^m$ satisfies the assignment for the circuit C with $u = (a_1, \ldots, a_{\ell_\phi})$, then the extractor returns the witness $w = (a_{\ell_\phi + 1}, \ldots, a_m)$.

With overwhelming probability, the extracted polynomial $v(x) := \sum_{i=0}^{d} c_i x^i$ does indeed provide a valid witness w. Otherwise, there exists a reduction to q-PDH that uses the SNARK adversary $\mathcal{A}^{\Pi}$. Define the polynomial:

$$v_{\mathsf{mid}}(x) := v(x) - v_0(x) - \sum_{i=1}^{\ell_\phi} a_i v_i(x).$$

We know by definition of SSP and by Theorem 4.2 that C is satisfiable if and only if:

$$t(x) \mid v^2(x) - 1 \ \wedge$$
$$v_{\mathsf{mid}}(x) = \sum_i^d c_i x^i - v_0(x) - \sum_i^{\ell_\phi} a_i v_i(x) \in \mathrm{Span}(v_{\ell_\phi+1}, \ldots, v_m, t)$$

Therefore, by contradiction, if the adversary $\mathcal{A}^{\Pi}$ does not know a witness $w \in \{0,1\}^{m-\ell_\phi}$ for u (such that $(u, w) \in \mathsf{R_C}$), but still the two verification checks Eq. (eq-div) and Eq. (eq-lin) pass, we have that either one of the following two cases must hold:

i. $t(x)h(x) \neq v^2(x) - 1$, but $t(s)h(s) = v^2(s) - 1$; or

ii. $v_{\mathsf{mid}}(x) \notin \mathrm{Span}(v_{\ell_\phi+1}, \ldots, v_m, t)$, but B_w is an encoding of $\beta v_{\mathsf{mid}}(s)$.

Let $\mathcal{B}^{\mathrm{PDH}}$ be an adversary against the q-PDH assumption. Given a q-PDH challenge:

$$\left(\mathsf{Enc.E}(1), \mathsf{Enc.E}(s), \ldots, \mathsf{Enc.E}(s^q), \mathsf{Enc.E}\left(s^{q+2}\right), \ldots, \mathsf{Enc.E}\left(s^{2q}\right)\right), \quad \text{for } q \in \{2d-1, d\}$$

adversary $\mathcal{B}^{\mathrm{PDH}}$ samples uniformly at random $\alpha \leftarrow\!\!\$\, \mathbb{F}$, and defines some $\beta \in \mathbb{F}$ (that we will formally construct later) and constructs a CRS as per Equation (4.4). There are some subtleties in how $\mathcal{B}^{\mathrm{PDH}}$ generates the value β. In fact, β can be generated without knowing its value explicitly, but rather knowing its representation over the power basis $\{s^i\}_{i=0, i\neq q+1}^{2q}$ – that is, knowing a polynomial $\beta(x)$ and its evaluation in s. Some particular choices of β will allow us to provide a solution for a q-PDH challenge. $\mathcal{B}^{\mathrm{PDH}}$ invokes the adversary $\mathcal{A}^{\Pi}$ as well as the extractor Ext^{Π} on the generated CRS, thus obtaining a proof π and the linear combination used by the prover for the polynomials $h(x), v(x)$ and also extracts a witness for the statement being proved.

For the *strong soundness* (see Remark 2.8), in order to simulate the verification oracle and to answer the verification queries of $\mathcal{A}^{\Pi}$, $\mathcal{B}^{\mathrm{PDH}}$ has to compare its encodings (obtained from the extracted coefficients and its input) with $\mathcal{A}$'s proof terms, accepts if the terms match, and rejects otherwise. Because the encoding scheme is not deterministic, adversary $\mathcal{B}^{\mathrm{PDH}}$ invokes the PKEQ extractor and simulates the verification oracle correctly with overwhelming probability.

The reduction in the two mentioned cases works as follows:

i. The extracted polynomials $h(x)$ and $v(x)$ satisfy $t(s)h(s) = v^2(s)-1$, but $t(x)h(x) \neq v^2(x)-1$. By q-PDH assumption this can happen only with negligible probability. We define $p(x) = v^2(x) - 1 - t(x)h(x)$, that in this case is a non-zero polynomial of degree $k \leq 2d$ having s as a root. Let p_k be the highest nonzero coefficient of $p(x)$. Write $\tilde{p}(x) = x^k - p_k^{-1} \cdot p(x)$. Since s is a root of $x^k - \tilde{p}(x)$, it is a root of $x^{q+1} - x^{q+1-k}\tilde{p}(x)$. $\mathcal{B}^{\mathrm{PDH}}$ solves q-PDH by computing $\mathsf{Enc.E}(s^{q+1}) = \mathsf{Enc.E}(s^{q+1-k}\tilde{p}(s))$ for $q = 2d-1$. Since $\deg(\tilde{p}) \leq k-1$, the latter is a known linear combination of encodings $\mathsf{Enc.E}(1), \mathsf{Enc.E}(s), \ldots, \mathsf{Enc.E}(s^q)$ which are available from the q-PDH challenge. More precisely, $\mathcal{B}^{\mathrm{PDH}}$ will compute $\mathsf{Eval}((\mathsf{Enc.E}(s^{i+q+1-k}))_i, (\tilde{p}_i)_i^{2d-1})$ on *fresh* encodings $\mathsf{Enc.E}(1), \mathsf{Enc.E}(s), \mathsf{Enc.E}(s^2), \ldots, \mathsf{Enc.E}(s^q)$ solving the q-PDH challenge for $q \geq 2d-1$.

ii. In the second case, suppose that the polynomial v_{mid} extracted as previously described cannot be expressed as a linear combination of $\{v_{\ell_\phi+1}, \ldots, v_m, t\}$. The proof still passes the verification, so we have a consistent value for $B_w \in [\mathsf{Enc.E}(\beta v_{\mathsf{mid}}(s))]$.

$\mathcal{B}^{\mathrm{PDH}}$ generates a uniformly random polynomial $a(x)$ of degree q subject to the constraint that all of the polynomials $a(x)t(x)$ and $\{a(x)v_i(x)\}_{i=\ell_\phi+1}^m$ have coefficient 0 for x^{q+1}. We note that for $q = d$, there are $q - (m - \ell_\phi) > 0$ degrees of freedom in choosing $a(x)$.

$\mathcal{B}^{\text{PDH}}$ defines β to be the evaluation of $a(x)$ in s, i.e., $\beta := a(s)$. Remark that $\mathcal{B}^{\text{PDH}}$ does not know s explicitly, but having access to the encodings of $2q-1$ powers of s, it is able to generate valid encodings $(\mathsf{Enc.E}(\beta v_i(s)))_i$ and $\mathsf{Enc.E}(\beta t(s))$ using Eval. Note that, by construction of β, this evaluation is of $d+1$ elements in $\mathbb{F}$ and that the $(q+1)$-th power of s is never used. Now, since $v_{\mathsf{mid}}(x)$ is not in the proper span, the coefficient of degree $q+1$ of $xa(x)v_{\mathsf{mid}}(x)$ must be nonzero with overwhelming probability $1-1/|\mathbb{F}|$. The term B_w of the proof must encode a known polynomial in s: $\sum_{i=0}^{2q} b_i s^i := \beta v_{\mathsf{mid}}(s) = a(s)v_{\mathsf{mid}}(s)$ where the coefficient b_{q+1} is non-trivial. $\mathcal{B}^{\text{PDH}}$ can subtract off encodings of multiples of other powers of s to recover $\mathsf{Enc.E}(s^{q+1})$ and break q-PDH. This requires an evaluation on *fresh* encodings:

$$\mathsf{Eval}\left((\mathsf{Enc.E}(s^i))_{\substack{i=0\\ i\neq q+1}}^{q+d}, (-b_i)_{\substack{i=0\\ i\neq q+1}}^{q+d}\right). \tag{4.8}$$

Adding the above to B_w and multiplying by the inverse of the $(q+1)$-th coefficient (using once again Eval) will provide a solution to the q-PDH problem for $q=d$.

Since the two cases above are not possible by q-PDH assumption, Ext^{Π} extracts a valid witness if the proof of $\mathcal{A}^{\Pi}$ is valid. □

Zero knowledge

In order to obtain a zero-knowledge protocol, we perform smudging of the proofs terms, and we randomize the target polynomial $t(x)$. The first step hides the witness, the second makes the distribution of the final noise independent from the coefficient a_i. The random vectors constituting the first element of the ciphertext are guaranteed to be statistically indistinguishable from uniformly random vectors by leftover hash lemma (cf. Lemma 4.17).

Proof of zero knowledge. The simulator for zero knowledge is shown in Figure 4.7. The error are independently sampled from the same uniform distribution over the (integer) interval $[-2^{\kappa}T\sigma_{B_w}, 2^{\kappa}T\sigma_{B_w}]$, where T is a small constant and $\sigma_{B_w} := p\sigma\sqrt{d+1}\sqrt{p^2+m-\ell_\phi}$. We will call this the *smudging distribution*.

Checking that the proof output by $\Pi.\mathsf{Sim}$ is indeed correct (i.e., that it verifies Eqs. (eq-pke) to (eq-lin)) is trivial. We are left with showing that the two proofs are statistically indistinguishable.

Note that once the value of V_w in the proof has been fixed, the verification equations uniquely determine $H, \hat{H}, \hat{V}$, and B_w. This means that for any (u,w) such that $\mathsf{C}(u,w)=1$, both the real arguments and the simulated arguments are chosen uniformly at random such that the verification equations will be satisfied. One can prove that values for V_w are statistically indistinguishable when executing $\Pi.\mathsf{P}$ and $\Pi.\mathsf{Sim}$: V_w is the encoding of a uniformly random variable γ_w in $\Pi.\mathsf{Sim}$ and the masking of a polynomial evaluation by adding $\gamma t(s)$, where γ is chosen uniformly at random (note that $t(s)\neq 0$) in $\Pi.\mathsf{P}$. What is encoded in the remaining terms is simply dictated by the verification constraints.

In both worlds, the proof is a tuple of 5 encodings $(H, \hat{H}, \hat{V}, V_w, B_w)$. Once the vrs is fixed, each encoding can be written as $(-\mathbf{a}, \mathbf{a}\cdot\mathbf{s}+pe+m)$, for some $\mathbf{a}\in\mathbb{Z}_q^n$ and some $m\in\mathbb{Z}_p$ satisfying the verification equations. Due to Lemma 4.16, the random vectors $\mathbf{a}$ are indistinguishable from uniformly random in both worlds. The error terms are statistically indistinguishable due to Lemma 4.12. (See Section 4.6 for a detailed explanation of these values.) □

Zero knowledge comes at a cost: smudging the error terms requires us to scale the ciphertext modulus by κ bits. For those applications where zero knowledge is not required, we can simplify the protocol by removing $\gamma t(x)$ from the computation of $h(x)$ and avoiding the smudging procedure on every proof term. In Table 4.1 we show some choices of parameters, both with and without zero knowledge.

Simulator $\Pi.\mathsf{Sim}(\sigma, \tau, \phi)$

$(\Gamma, \mathsf{sk}, s, \alpha, \beta) \coloneqq \tau$; $(a_1, \ldots, a_{\ell_\phi}) \coloneqq \phi$

$\gamma_w \leftarrow\!\!\$\, \mathbb{F}$

$h \coloneqq \left((v_0(s) + \sum_i^{\ell_\phi} a_i v_i(s) + \gamma_w)^2 - 1\right) \;/\; t(s)$

$H \leftarrow \mathsf{Enc.E}(h)$; $\hat{H} \leftarrow \mathsf{Enc.E}(\alpha h)$; $\hat{V} \leftarrow \mathsf{Enc.E}(\alpha v_0(s) + \sum_i^{\ell_\phi} a_i \alpha v_i(s) + \alpha\gamma_w)$

$V_w \leftarrow \mathsf{Enc.E}(\gamma_w)$; $B_w \leftarrow \mathsf{Enc.E}(\beta\gamma_w)$

Apply smudging on $H, \hat{H}, \hat{V}, B_w, V_w$

return $(H, \hat{H}, \hat{V}, V_w, B_w)$

Figure 4.7: Simulator for zero knowledge.

Knowledge soundness

Before diving into the technical details of the proof of soundness, we provide some intuition in an informal sketch of the security reductions: the CRS for the scheme contains encodings of $\mathsf{Enc.E}(s)$, $\ldots, \mathsf{Enc.E}(s^d)$, as well as encodings of these terms multiplied by some field elements $\alpha, \beta \in \mathbb{F}$. The scheme requires the prover P to exhibit encodings computed homomorphically from such CRS.

The reason for requiring the prover to duplicate its effort w.r.t. α is so that the simulator in the security proof can extract representations of $\hat{V}$, $\hat{H}$ as degree-d polynomials $v(x)$, $h(x)$ such that $v(s) = v_s, h(s) = h_s$, by the q-PKE assumption (for $q = d$). The assumption also guarantees that this extraction is efficient. This explains the first quadratic root detection check Equation (eq-pke) in the verification algorithm.

Suppose an adversary manages to forge a SNARK of a false statement and pass the verification test. Then, by soundness of the square span program (Theorem 4.2), for the extracted polynomials $v(x), h(x)$ and for the new defined polynomial $v_{\mathsf{mid}}(x) \coloneqq v(x) - v_0(x) - \sum_i^{\ell_\phi} a_i v_i(x)$, one of the following must be true:

i. $h(x)t(x) \neq v^2(x) - 1$, but $h(s)t(s) = v^2(s) - 1$, from Equation (eq-div);

ii. $v_{\mathsf{mid}}(x) \notin \mathrm{Span}(v_{\ell_\phi+1}, \ldots, v_m)$, but B_w is a valid encoding of $\mathsf{Enc.E}(\beta v_{\mathsf{mid}}(s))$, from Equation (eq-lin).

If the first case holds, then $p(x) \coloneqq (v^2(x) - 1) - h(x)t(x)$ is a nonzero polynomial of degree some $k \leq 2d$ that has s as a root, since the verification test implies $(v^2(s) - 1) - h(s)t(s) = 0$. The simulator can use $p(x)$ to solve q-PDH for $q \geq 2d - 1$ using the fact that $\mathsf{Enc.E}\left(s^{q+1-k}p(s)\right) \in [\mathsf{Enc.E}(0)]$ and subtracting off encodings of lower powers of s to get $\mathsf{Enc.E}(s^{q+1})$.

To handle the second case, i.e., to ensure that $v_{\mathsf{mid}}(x)$ is in the linear span of the $v_i(x)$'s with $\ell_\phi < i \leq m$ we use an extra scalar β, supplement the CRS with the terms $[\mathsf{Enc.E}(\beta v_i(s))]_{i>\ell_\phi}$, $\mathsf{Enc.E}(\beta t(s))$, and require the prover to present (encoded) $\beta v_{\mathsf{mid}}(s)$ in its proof. The adversary against q-PDH will choose a polynomial $\beta(x)$ convenient to solve the given instance. More specifically, it sets $\beta(x)$ with respect to the set of polynomials $\{v_i(x)\}_{i>\ell_\phi}$ such that the coefficient for x^{q+1} in $\beta(x)v_{\mathsf{mid}}(x)$ is zero. Then, to generate the values in the σ it sets $\beta \coloneqq \beta(s)$ (which can be computed from its input consisting of encodings of powers of s). Using the above, it runs the SNARK adversary and to obtain from its output B_w an encoding of some polynomial with coefficient s^{q+1} non-zero and thus solve q-PDH. Also here, the verification algorithm guarantees that even with all the above homomorphic operations, the challenger still decrypts the correct value with $1 - \mathsf{negl}(\kappa)$ probability.

As previously mentioned in Remark 2.8, the proof of knowledge soundness allows oracle access to the verification procedure. In the context of a weaker notion of soundness where the adversary does not have access to the $\Pi.\mathsf{V}(\mathsf{vrs}, \cdot, \cdot)$ oracle, the proof is almost identical, except that there is no need for the $\mathcal{B}^{\mathrm{PDH}}$ adversary to answer queries and to simulate the verification, and therefore no need for the q-PKEQ assumption. This greatly simplifies our construction: the protocol does not need to rely on the q-PKEQ assumption, and the prime modulus can be of κ bits.

4.6 Efficiency and concrete parameters

The prover's computations are bounded by the security parameter and the size of the circuit, i.e., $\mathsf{P} \in \widetilde{O}(\lambda d)$. As in [GGPR13, DFGK14], the verifier's computations depend solely on the security parameter, i.e., $\mathsf{V} \in O(\lambda)$. The proof consists of a constant number (precisely, 5) of LWE encodings, i.e., $|\pi| = 5 \cdot \widetilde{O}(\lambda)$. Finally, the complexity for the setup procedure is $\widetilde{O}(\lambda d)$.

Using the propositions from Section 4.2 and knowing the exact number of homomorphic operations that need to be performed in order to produce a proof, we can now attempt at providing some concrete parameters for our encoding scheme.

We fix the statistical security parameter $\kappa := 32$, as already done in past works on fully homomorphic encryption (e.g., [DM15, CGGI16]). We fix the circuit size $d := 2^{15}$, which is sufficient for some practical applications such as the computation of SHA-256. For some practical examples of circuits, we direct the reader towards [BCG+14, PHGR13].

For a first attempt at implementing our solution, we assume a weaker notion of soundness, i.e., that in the KSND game the adversary does not have access to a verification oracle (cf. Figure 2.2). Concretely, this means that the only bound in the size of p is given by the guessing probability of the witness, and the guessing of a field element. We thus fix p to be a prime[3] of 32 bits for the size of the message space.

The CRS is composed of encodings of different nature: some of them are fresh ($\mathsf{Enc.E}(1)$, $\mathsf{Enc.E}(s), \ldots, \mathsf{Enc.E}(s^d)$), some happen to be stale in the construction of $\mathcal{A}^{\mathrm{PKE}}$ and the construction of $\mathcal{B}^{\mathrm{PDH}}$ Section 4.5 (Item i.) ($\mathsf{Enc.E}(\alpha s), \ldots, \mathsf{Enc.E}(\alpha s^d)$), and some are stale from the construction of $\mathcal{B}^{\mathrm{PDH}}$ Section 4.5 (Item ii.) ($\mathsf{Enc.E}(\beta t(s)), (\mathsf{Enc.E}(\beta v_i(s)))_i$). Since, as we have seen, $\mathcal{B}^{\mathrm{PDH}}$ manipulates the q-PDH challenge via homomorphic operations, we must guarantee that the protocol adversary can perform *at least* the same number of homomorphic operations as in the real-world protocol. Therefore, in the real protocol, we must intentionally increase the magnitude of the noise in the CRS: the terms $\mathsf{Enc.E}(\alpha s^i)$ (with $i = 0, \ldots, d$) are generated by multiplying the respective *fresh* encoding $\mathsf{Enc.E}(s^i)$ by a term bounded by p; the terms $\mathsf{Enc.E}(\beta t(s)), [\mathsf{Enc.E}(\beta v_i(s))]_i$ instead are generated via Eval of $d+1$ elements with coefficients bounded by p. Concretely, when encoding these elements using the encoding scheme of Section 4.2, the error for $\mathsf{Enc.E}(\alpha s^i)$ is sampled from $p \cdot \chi_\sigma$; the error for $\mathsf{Enc.E}(\beta t(s)), \mathsf{Enc.E}(\beta v_i(s)))$ is sampled from $(p\sqrt{d+1}) \cdot \chi_\sigma$.

The proof π consists of five elements $(H, \hat{H}, \hat{V}, V_w, B_w)$, as per Equation (4.6). H and V_w are computed using an affine function on d encodings with coefficients modulo p; $\hat{H}, \hat{V}$ are computed using a linear function on $d+1$ encodings with coefficients modulo p; finally, B_w is computed using a linear combination of $m - \ell_\phi$ encodings with coefficients in $\{0, 1\}$, except the last one which is modulo p. Overall, the term that carries the highest load of homomorphic computations is B_w. To it, we add a smudging term for constructing a zero knowledge proof π.

In the construction of the adversary $\mathcal{B}^{\mathrm{PDH}}$ (Item ii.) we need to perform some further homomorphic operations on the proof element B_w in order to solve the q-PDH challenge, namely one addition (Equation (4.8)) and one multiplication by a known scalar b bounded by p. The final result is the solution to the q-PDH challenge.

[3] In particular, we need p and q to be relatively prime for the correctness of the encoding scheme [BV11, footnote 18].

We now outline the calculations that we use to choose the relevant parameters for our encoding scheme. In particular, we will focus on the term B_w since, as already stated, it is the one that is involved in the largest number of homomorphic operations. The correctness of the other terms follows directly from Corollary 4.11.

First of all, the terms $(\beta v_i(s))_i$ and $\beta t(s)$ are produced through the algorithm Eval executed on $d+1$ *fresh* encodings with coefficients modulo p. Let σ be the discrete Gaussian parameter of the noise terms in fresh encodings; then, by Pythagorean additivity, the Gaussian parameter of encodings output by this homomorphic evaluation is $\sigma_{\mathsf{Eval}} \coloneqq p\sigma\sqrt{d+1}$. Then the term $\beta t(s)$ is multiplied by a coefficient in $\mathbb{Z}_p$, and the result is added to a subset sum of the terms $(\beta v_i(s))_i$, i.e., a weighted sum with coefficients in $\{0,1\}^\lambda$. It is trivial to see that, for the first term, the resulting Gaussian parameter is bounded by $p\sigma_{\mathsf{Eval}}$, whereas for the second term it is bounded by $\sigma_{\mathsf{Eval}}\sqrt{m-\ell_\phi}$. The parameter of the sum of these two terms is then bounded by $\sigma_{B_w} \coloneqq \sigma_{\mathsf{Eval}}\sqrt{p^2+m-\ell_\phi}$. Let us then consider a constant factor T for "cutting the Gaussian tails", i.e., such that the probability of sampling from the distribution and obtaining a value with magnitude larger than T times the standard deviation is as small as desired. We can then write that the absolute value of the error in B_w is bounded by $T\sigma_{B_w}$. At this point we add a *smudging* term, which amounts to multiplying the norm of the noise by $(2^\kappa+1)$ (cf. Corollary 4.13). Finally, the so-obtained encoding has to be summed with the output of an Eval invoked on $2d$ fresh encodings with coefficients modulo p and multiplied by a constant in $\mathbb{Z}_p$. The final noise is then bounded by $Tp\sigma_{B_w}(2^\kappa+1)+Tp\sigma_{\mathsf{Eval}}$ (cf. Lemma 4.14). By substituting the values of σ_{Eval}, σ_{B_w}, remembering that $\sigma \coloneqq \alpha q$ and imposing the condition for having a valid encoding, we obtain:

$$Tp^2\alpha q\sqrt{d+1}\left(\sqrt{p^2+m-\ell_\phi}\,(2^\kappa+1)+1\right) < \frac{q}{2p}.$$

The above corresponds to Equation (4.3) with bounds $B_e \coloneqq T\sigma_{B_w}$ and $B_{\mathsf{Eval}} \coloneqq T\sigma_{\mathsf{Eval}}$. By simplifying q and isolating α, we get:

$$\alpha < \frac{1}{2Tp^3\sqrt{d+1}\left(\sqrt{p^2+m-\ell_\phi}\,(2^\kappa+1)+1\right)}.$$

With our choice of parameters and by taking $T=8$, we can select for instance $\alpha = 2^{-180}$.

Once α and p are chosen, we select the remaining parameters q and n in order to achieve the desired level of security for the LWE encoding scheme. To do so, we take advantage of Albrecht's estimator[4] [APS15] which, as of now, covers the following attacks: meet-in-the-middle exhaustive search, coded-BKW [GJS15], dual-lattice attack and small/sparse secret variant [Alb17], lattice reduction with enumeration [LP11], primal attack via uSVP [AFG14, BG14], Arora-Ge algorithm [AG11] using Gröbner bases [ACFP14]. Some possible choices of parameters are reported in Table 4.1.

Finally, based on these parameters, we can concretely compute the size of the CRS[5] and that of the proof π. The CRS is composed of $d+(d+1)+(m+1)$ encodings, corresponding to the encodings of the d powers of s, the encodings of α multiplied by the $d+1$ powers of s, the m encodings of $(\beta v_i)_i$, and the encoding of $\beta t(s)$. This amounts to $(2d+m+2)$ LWE encodings, each of which has size $(n+1)\log q$ bits[6]. For the calculations, we bound m by d and state that the size of the CRS is that of $(3d+2)$ LWE encodings. From an implementation point of view,

[4] https://bitbucket.org/malb/lwe-estimator

[5] We take into account only the encodings that are contained in the CRS. The other terms have considerably smaller impact on its size or can be agreed upon offline (e.g., the SSP).

[6] Note that the magnitude of the noise term, i.e., whether the encoding is fresh or stale, has no impact on the size of an encoding. This size is a function only of n (the number of elements in the vector) and the modulus q.

Table 4.2: Comparison with previous works. PQ stands for post-quantum. We note that the construction of [PHGR13] is very different (namely, based on elliptic curves), and comparing security levels is therefore difficult.

	PQ	λ	ZK	$\lvert\pi\rvert$	$\lvert\sigma\rvert$	d
[PHGR13]	✗	256	✓	288 B	6.50 MB	23,785
[BISW17]	✓	100	✗	0.02 MB	1.23 GB	10,000
Section 4.4	✓	162	✓	0.64 MB	8.63 MB	32,767

we can consider LWE encodings $(\mathbf{a}, b) \in \mathbb{Z}_q^{n+1}$ where the vector $\mathbf{a}$ is the output of a seeded PRG. This has been proven secure in the random oracle model [Gal13]. Therefore, the communication complexity is greatly reduced, as sending an LWE encoding just amounts to sending the seed for the PRG and the value $b \in \mathbb{Z}_q$. For security to hold, we can take the size of the seed to be λ bits, thus obtaining the final size of the CRS: $(3d + 2) \log q + \lambda$ bits. The proof π is composed of 5 LWE encodings, therefore it has size $|\pi| = 5\,(n + 1) \log q$ bits. Note that in this case we cannot trivially use the same trick with the PRG, since the encodings are produced through homomorphic evaluations.

In Table 4.2 we show a comparison between our implementation, the zk-SNARK of [PHGR13] (informally called "Pinocchio"), and the recent implementation of [BISW17] by Samir Menon, Brennan Shacklett, and David Wu[7]. Despite the fact that the construction of Parno et al. [PHGR13] is fundamentally different as it targets encoding over elliptic curves, we believe that they provide a good term of comparison (when used with circuits of the same size) for the loss incurred when using lattice-based encodings instead. Note therefore that the security parameter of [PHGR13] is not comparable with the two other results.

Moreover, it is worth noting that the implementation of [BISW17] targets 80 bits of security, which is justified using the estimate provided in [LP11]. We report $\lambda = 100$ as given by Albrecht's tool [APS15], which we believe to be more accurate. Nonetheless, the estimated post-quantum security level is 50, thus insufficient for modern applications. Additionally, we note that, despite targeting the construction of SNARGs, it seems the construction of [BISW17] can be turned into a SNARK by using the stronger extractable linear-only assumption. In order to achieve this, they can use a technique called *double encryption*, which doubles the size of each ciphertext. More details about this are given in Section 4.3.

Implementation

We implemented our construction in standard C11, using the library GMP [Gt12] for handling arbitrary precision integers and the library FLINT [HJP13] for handling polynomials. We chose the pseudo-Mersenne prime $p := 2^{32} - 5$, and a the modulus $q := 2^{736}$. This allows for fast arithmetic operations: reduction modulo q simply consists in a bitmask, modular operations by p can fit a `uint64_t` type, and multiplication of a scalar modulo p to a vector in $\mathbb{Z}_q^{n+1}$ does not require any memory allocation for the carry. The dimension of the lattice was chosen $n = 1470$, corresponding to the "medium" security level displayed in Table 4.1. We used AES-256 in counter mode as a PRG, taking advantage of AES-NI instructions when available.

We performed extensive benchmarks of our protocol on a single thread of an Intel Core i7-4770K CPU @ 3.50GHz, running Debian (kernel version 4.9.110). Our implementation is publicly

[7]Results are extracted from the source code at https://github.com/dwu4/lattice-snarg.

Table 4.3: Benchmarks of our proof system (zk) for different circuit sizes (i.e., d).

Circ. size	Setup (s)	Prover (s)	Verifier (ms)
2^{10}	1.46 s ± 18.7 ms	1.61 s ± 27.8 ms	1.26 ms ± 16 µs
2^{13}	12.3 s ± 37.9 ms	13 s ± 224 ms	1.50 ms ± 16 µs
2^{15}	57.8 s ± 134 ms	53.6 s ± 247 ms	2.28 ms ± 17 µs
2^{16}	167 s ± 269 ms	235 s ± 451 ms	3.46 ms ± 17 µs

available [8]. Time is measured using `gettimeofday(2)`. Encoding a uniformly random element of $\mathbb{Z}_p$ using Enc.Enc.E takes on average 310 µs (std. dev. 34 µs); decoding it using Enc.Enc.D is about the same order of magnitude, 197 µs (std. dev. 24 µs). Measurements were done over 100,000 samples. The algorithm for homomorphic evaluations Eval is able to compute a linear combination of 2^{15} ciphertexts with coefficients modulo p in roughly 13 s, and of 2^{18} in about 94 s.

For proving satisfiability of a boolean circuit with roughly 2^{14} gates (i.e., $d = 2^{15}$), we measured 57 s for the CRS generation algorithm; 53 s for the prover; 2.28 ms for the verifier (on average, over 100 repeated executions varying SSPs). This is about one order of magnitude slower w.r.t. Pinocchio's benchmarks [PHGR13, Fig. 8]; verification instead is one order of magnitude faster. More detailed benchmarks for different circuit sizes can be found in Table 4.3.

[8] See https://www.di.ens.fr/~orru/pq-zk-snarks.

Chapter 5

Mimblewimble: a private cryptocurrency

This work was published in the proceedings of the 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT. It was completed with co-authors Georg Fuchsbauer and Yannick Seurin. In this particular work, my role was largely assistive, with Georg and Yannick leading the way.

Proposed in 2008 and launched early 2009, Bitcoin [Nak08] is a decentralized payment system in which transactions are registered in a distributed and publicly verifiable ledger called a blockchain. Bitcoin departs from traditional account-based payment systems where transactions specify an amount moving from one account to another. Instead, each transaction consists of a list of *inputs* and a list of *outputs*. Each output contains a value (expressed as a multiple of the currency unit, 10^{-8} bitcoin) and a short script specifying how the output can be spent. The most common script is *Pay to Public Key Hash* (P2PKH) and contains the hash of an ECDSA public key, commonly called a Bitcoin address. Each input of a transaction contains a reference to an output of a previous transaction in the blockchain and a script that must match the script of that output. In the case of P2PKH, an input must provide a public key that hashes to the address of the output it spends and a valid signature for this public key.

Each transaction spends one or more previous transaction outputs and creates one or more new outputs, with a total value not larger than the total value of coins being spent. The system is bootstrapped through special transactions called *coinbase* transactions, which have outputs but no inputs and therefore create money (and also serve to incentivize the proof-of-work consensus mechanism, which allows users to agree on the valid state of the blockchain).

To avoid double-spending attacks, each output of a transaction can only be referenced once by an input of a subsequent transaction. Note that this implies that an output must necessarily be spent entirely. As transactions can have multiple outputs, change can be realized by having the sender assign part of the outputs to an address she controls. Since all transactions that ever occurred since the inception of the system are publicly available in the blockchain, whether an output has already been spent can be publicly checked. In particular, every transaction output recorded in the blockchain can be classified either as an *unspent transaction output (UTXO)* if it has not been referenced by a subsequent transaction input so far, or a *spent transaction output (STXO)* otherwise. Hence, the UTXO set "encodes" all bitcoins available to be spent, while the STXO set only contains "consumed" bitcoins and could, in theory, be deleted.

The validation mechanics in Bitcoin requires new users to download and validate the entire

blockchain in order to check that their view of the system is not compromised.[1] Consequently, the security of the system and its ability to enroll new users relies on (a significant number of) Bitcoin clients to persistently store the entire blockchain. Once a new node has checked the entire blockchain, it is free to "prune" it[2] and retain only the freshly computed UTXO set, but it will not be able to convince another newcomer that this set is valid.

Consider the following toy example. A coinbase transaction creates an output txo_1 for some amount v associated with a public key pk_1. This output is spent by a transaction T_1 creating a new output txo_2 with amount v associated with a public key pk_2. Transaction T_1 contains a valid signature σ_1 under public key pk_1. Once a node has verified σ_1, it is ensured that txo_2 is valid and the node can therefore delete the coinbase transaction and T_1. By doing this, however, he cannot convince anyone else that output txo_2 is indeed valid.

At the time of writing, the size of Bitcoin's blockchain is over 200 GB.[3] Downloading and validating the full blockchain can take up to several days on standard hardware. In contrast, the size of the UTXO set, containing around 60 millions elements, is only a couple of GB.

Bitcoin privacy. Despite some common misconception, Bitcoin offers a very weak level of privacy. Although users can create multiple pseudonymous addresses at will, the public availability of all transaction data often allows to link them and reveals a surprisingly large amount of identifying information, as shown in many works [AKR+13, MPJ+13, RS13, KKM14].

Several protocols have been proposed with the goal of improving on Bitcoin's privacy properties, such as Cryptonote [vS13] (implemented for example by Monero), Zerocoin [MGGR13] and Zerocash [BCG+14]. On the other hand, there are privacy-enhancing techniques compatible with Bitcoin, for example coin mixing [BBSU12, BNM+14, RMK14, HAB+17], to ensure payer anonymity. Below we describe three specific proposals that have paved the way for Mimblewimble.

Confidential Transactions. Confidential Transactions (CT), described by Maxwell [Max15], based on an idea by Back [Bac13] and now implemented by Monero, allow to hide the *values* of transaction outputs. The idea is to replace explicit amounts in transactions by homomorphic commitments: this hides the value contained in each output, but the transaction creator cannot modify this value later on.[4]

More specifically, the amount v in an output is replaced by a Pedersen commitment $C = vH + rG$, where H and G are generators of an (additively denoted) discrete-log-hard group and r is a random value. Using the homomorphic property of the commitment scheme, one can prove that a transaction does not create money out of thin air, i.e., that the sum of the outputs is less than the sum of the inputs. Consider a transaction with input commitments $C_i = v_i H + r_i G$, $1 \leq i \leq n$, and output commitments $\hat{C}_i = \hat{v}_i H + \hat{r}_i G$, $1 \leq i \leq m$. The transaction does not create money *iff* $\sum_{i=1}^{n} v_i \geq \sum_{i=1}^{m} \hat{v}_i$. This can be proved by providing an opening (f, r) with $f \geq 0$ for $\sum_{i=1}^{n} C_i - \sum_{i=1}^{m} \hat{C}_i$, whose validity can be publicly checked. The difference f between inputs and outputs are so-called fees that reward the miner that includes the transaction in a block.

Note that arithmetic on hidden values is done modulo p, the order of the underlying group. Hence, a malicious user could spend an input worth 2 and create two outputs worth 10 and $p - 8$, which would look the same as a transaction creating two outputs worth 1 each. To ensure that

[1] So-called *Simplified Verification Payment (SPV)* clients only download much smaller pieces of the blockchain allowing them to verify specific transactions. However, they are less secure than fully validating clients and they do not contribute to the general security of the system [GCKG14, SZ16].

[2] This functionality was introduced in Bitcoin Core v0.11, see https://github.com/bitcoin/bitcoin/blob/v0.11.0/doc/release-notes.md#block-file-pruning.

[3] See https://www.blockchain.com/charts/blocks-size.

[4] Commitments are actually never publicly opened; however the opening information is used when spending a coin and remains privy to the participants.

commitments do not contain large values that cause such mod-p reductions, a non-interactive zero-knowledge (NIZK) proof that the committed value is in $[\![0, v_{\max}]\!]$ (a so-called *range proof*) is added to each commitment, where $v_{\max}$ is small compared to p.

CoinJoin. When a Bitcoin transaction has multiple inputs and outputs, nothing can be inferred about "which input goes to which output" beyond what is imposed by their values (e.g., if a transaction has two inputs with values 10 BTC and 1 BTC, and two outputs with values 10 BTC and 1 BTC, all that can be said is that at least 9 BTC flowed from the first input to the first output). CoinJoin [Max13a] builds on this technical principle to let different users create a single transaction that combines all of their inputs and outputs. When all inputs and outputs have the same value, this perfectly mixes the coins. Note that unlike CT, CoinJoin does not require any change to the Bitcoin protocol and is already used in practice. However, this protocol is interactive as participants need all input and output addresses to build the transaction. Saxena et al. [SMD14b] proposed a modification of the Bitcoin protocol which essentially allows users to perform CoinJoin non-interactively and which relies on so-called *composite* signatures.[5]

Cut-through. A basic property of the UTXO model is that a sequence of two transactions, a first one spending an output txo_1 and creating txo_2, followed by a second one spending txo_2 and creating txo_3, is equivalent to a single *cut-through* transaction spending txo_1 and creating txo_3. While such an optimization is impossible once transactions have been included in the blockchain (as mentioned before, this would violate public verifiability of the blockchain), this has been suggested [Max13b] for *unconfirmed* transactions, i.e., transactions broadcast to the Bitcoin network but not included in a block yet. As we will see, the main added benefit of Mimblewimble is to allow *post-confirmation cut-through*.

Mimblewimble. Mimblewimble was first proposed by an anonymous author in 2016 [Jed16]. The idea was then developed further by Poelstra [Poe16]. At the time of writing, there are at least two independent implementations of Mimblewimble as a cryptocurrency: one is called Grin,[6] the other Beam.[7]

Mimblewimble combines in a clever way CT, a non-interactive version of CoinJoin, and cut-through of transaction inputs and outputs. As with CT, a coin is a commitment $C = vH + rG$ to its value v using randomness r, together with a range proof π. If CT were actually employed in Bitcoin, spending a CT-protected output would require the knowledge of the opening of the commitment *and*, as for a standard output, of the secret key associated with the address controlling the coin. Mimblewimble goes one step further and completely abandons the notion of addresses or more generally scripts: spending a coin *only* requires knowledge of the opening of the commitment. As a result, ownership of a coin $C = vH + rG$ is equivalent to the knowledge of its opening, and the randomness r of the commitment now acts as the *secret key* for the coin.

Exactly as in Bitcoin, a Mimblewimble transaction specifies a list $\mathbf{C} = (C_1, \ldots, C_n)$ of input coins (which must be coins existing in the system) and a list $\hat{\mathbf{C}} = (\hat{C}_1, \ldots, \hat{C}_m)$ of output coins, where $C_i = v_i H + r_i G$ for $1 \leq i \leq n$ and $\hat{C}_i = \hat{v}_i H + \hat{r}_i G$ for $1 \leq i \leq m$. We will detail later how exactly such a transaction is constructed. Leaving fees aside for simplicity, the transaction is balanced (i.e., does not create money) *iff* $\sum \hat{v}_i - \sum v_i = 0$, which, letting $\sum \mathbf{C}$ denote $\sum_{i=1}^{n} C_i$, is

[5] An earlier, anonymous version of the paper used the name *one-way aggregate signature* (OWAS), see https://bitcointalk.org/index.php?topic=290971. Composite signatures are very similar to aggregate signatures [BGLS03].

[6] See http://grin-tech.org and https://github.com/mimblewimble/grin/blob/master/doc/intro.md.

[7] See https://www.beam-mw.com.

equivalent to:

$$\sum \hat{\mathbf{C}} - \sum \mathbf{C} = \sum(\hat{v}_i H + \hat{r}_i G) - \sum(v_i H + r_i G) = \left(\sum \hat{r}_i - \sum r_i\right) G \,.$$

In other words, knowledge of the opening of all coins in the transaction *and* balancedness of the transaction implies knowledge of the discrete logarithm in base G of $E \coloneqq \sum \hat{\mathbf{C}} - \sum \mathbf{C}$, called the *excess* of the transaction in Mimblewimble jargon. Revealing the opening $(0, r \coloneqq \sum \hat{r}_i - \sum r_i)$ of the excess E as in CT would leak too much information (e.g., together with the openings of the input coins and of all output coins except one, this would yield the opening of the remaining output coin); however, knowledge of r can be *proved* by providing a valid signature (on the empty message) under public key E using some discrete-log-based signature scheme. Intuitively, as long as the commitment scheme is binding and the signature scheme is unforgeable, it should be infeasible to compute a valid signature for an unbalanced transaction.

Transactions (legitimately) creating money, such as coinbase transactions, can easily be incorporated by letting the *supply* s (i.e., the number of monetary units created by the transaction) be explicitly specified and redefining the excess of the transaction as $E \coloneqq \sum \hat{\mathbf{C}} - \sum \mathbf{C} - sH$. All in all, a Mimblewimble transaction is a tuple

$$\mathsf{tx} = (s, \mathbf{C}, \hat{\mathbf{C}}, K) \quad \text{with} \quad K \coloneqq (\pi, \mathbf{E}, \sigma) \,, \tag{5.1}$$

where s is the supply, $\mathbf{C}$ is the input coin list, $\hat{\mathbf{C}}$ is the output coin list, and K is the so-called *kernel*, which contains the list π of range proofs for output coins,[8] the (list of) transaction excesses $\mathbf{E}$ (as there can be several; see below), and a signature σ.[9]

Such transactions can now easily be merged non-interactively *à la* CoinJoin: consider $\mathsf{tx}_0 = (s_0, \mathbf{C}_0, \hat{\mathbf{C}}_0, (\pi_0, E_0, \sigma_0))$ and $\mathsf{tx}_1 = (s_1, \mathbf{C}_1, \hat{\mathbf{C}}_1, (\pi_1, E_1, \sigma_1))$; then the *aggregate transaction* tx resulting from merging tx_0 and tx_1 is simply

$$\mathsf{tx} \coloneqq \left(s_0 + s_1, \mathbf{C}_0 \parallel \mathbf{C}_1, \hat{\mathbf{C}}_0 \parallel \hat{\mathbf{C}}_1, (\pi_0 \parallel \pi_1, (E_0, E_1), (\sigma_0, \sigma_1))\right) . \tag{5.2}$$

Moreover, if the signature scheme supports aggregation, as for example the BLS scheme [BGLS03, BNN07], the pair (σ_0, σ_1) can be replaced by a compact aggregate signature σ for the public keys $\mathbf{E} \coloneqq (E_0, E_1)$.

An aggregate transaction $(s, \mathbf{C}, \hat{\mathbf{C}}, (\pi, \mathbf{E}, \sigma))$ is valid if all range proofs verify, σ is a valid aggregate signature for $\mathbf{E}$ and if

$$\sum \hat{\mathbf{C}} - \sum \mathbf{C} - sH = \sum \mathbf{E} \,. \tag{5.3}$$

As transactions can be recursively aggregated, the resulting kernel will contain a list $\mathbf{E}$ of kernel excesses, one for each transaction that has been aggregated.

The main novelty of Mimblewimble, namely cut-through, naturally emerges from the way transactions are aggregated and validated. Assume that some coin C appears as an output in tx_0 and as an input in tx_1; then, one can erase C from the input and output lists of the aggregate transaction tx, and tx will still be valid since (5.3) will still hold. Hence, each time an output of a transaction tx_0 is spent by a subsequent transaction tx_1, this output can be "forgotten" without losing the ability to validate the resulting aggregate transaction.

In Mimblewimble the ledger is itself a transaction of the form (5.1), which starts out empty, and to which transactions are recursively aggregated as they are added to the ledger. We assume that for a transaction to be allowed onto the ledger, its input list must be contained in the output list of the ledger (this corresponds to the natural requirement that only coins that exist in the ledger can be spent). Then, it is easy to see that the following holds:

[8] Since inputs must be coins that already exist in the system, their range proofs are contained in the kernels of the transactions that created them.

[9] A transaction fee can easily be added to the picture by making its amount f explicit and adding fH to the transaction excess. For simplicity, we omit it here.

(i) the supply s of the ledger is equal to the sum of the supplies of all transactions added to the ledger so far;

(ii) the input coin list of the ledger is always empty.

Property (i) follows from the definition of aggregation in (5.2). Property (ii) follows inductively. At the inception of the system the ledger is empty (thus the first transaction added to the ledger must be a transaction with an empty input coin list and non-zero supply, a *minting* transaction). Any transaction tx added to the ledger must have its input coins contained in the output coin list of the ledger; thus cut-through will remove all of them from the joint input list, hence the updated ledger again has no input coins (and the coins spent by tx are deleted from its outputs). The ledger in Mimblewimble is thus a single aggregate transaction whose supply s is equal to the amount of money that was created in the system and whose output coin list $\hat{\mathbf{C}}$ is the analogue of the UTXO set in Bitcoin. Its kernel K allows to cryptographically verify its validity. The history of all transactions that have occurred is not retained, and only one kernel excess per transaction (a very short piece of information) is recorded.

Our contribution. A first attempt at proving the security of Mimblewimble was partly undertaken by Poelstra [Poe16]. We follow a different approach: we put forward a general syntax and a framework of game-based security definitions for an abstraction of Mimblewimble that we dub an *aggregate cash system*.

Formalizing security for a cash system requires care. For example, Zerocoin [MGGR13] was recently found to be vulnerable to *denial-of-spending* attacks [RTRS18] that were not captured by the security model in which Zerocoin was proved secure. To avoid such pitfalls, we keep the syntax simple, while allowing to express meaningful security definitions. We formulate two natural properties that define the security of a cash system: *inflation-resistance* ensures that the only way money can be created in a system is explicitly via the supply contained in transactions; *resistance to coin theft* guarantees that no one can spend a user's coins as long as she keeps her keys safe. We moreover define a privacy notion, *transaction indistinguishability*, which states that a transaction does not reveal anything about the values it transfers from its inputs to its outputs.

We then give a black-box construction of an aggregate cash system, which naturally generalizes Mimblewimble, from a homomorphic commitment scheme Com, an (aggregate) signature scheme Sig, and a NIZK range-proof system Π. We believe that such a modular treatment will ease the exploration of post-quantum instantiations of Mimblewimble or related systems.

Note that in our description of Mimblewimble, we have not yet explained how to actually create a transaction that transfers some amount ρ of money from a sender to a receiver. It turns out that this is a delicate question. The initial description of the protocol [Jed16] proposed the following one-round procedure:

- the sender selects input coins $\mathbf{C}$ of total value $v \geq \rho$; it creates *change coins* $\mathbf{C}'$ of total value $v - \rho$ and sends $\mathbf{C}$, $\mathbf{C}'$, range proofs for $\mathbf{C}'$ and the opening $(-\rho, k)$ of $\sum \mathbf{C}' - \sum \mathbf{C}$ to the receiver (over a secure channel);
- the receiver creates additional output coins $\mathbf{C}''$ (and range proofs) of total value ρ with keys (k_i''), computes a signature σ with the secret key $k + \sum k_i''$ and defines the transaction $\mathsf{tx} = \big(0, \mathbf{C}, \mathbf{C}' \,\|\, \mathbf{C}'', (\pi, E = \sum \mathbf{C}' + \sum \mathbf{C}'' - \sum \mathbf{C}, \sigma)\big)$.

However, a subtle problem arises with this protocol. Once the transaction has been added to the ledger, the change outputs $\mathbf{C}'$ should only be spendable by the sender, who owns them. It turns out that the receiver is also able to spend them by "reverting" the transaction tx. Indeed, he knows the range proofs for coins in $\mathbf{C}$ and the secret key $(-k - \sum k_i'')$ for the transaction with inputs

$\mathbf{C}' \parallel \mathbf{C}''$ and outputs $\mathbf{C}$. Arguably, the sender is given back her initial input coins in the process, but (i) she could have deleted the secret keys for these old coins, making them unspendable, and (ii) this violates any meaningful formalization of security against coin theft.

A natural way to prevent such a malicious behavior would be to let the sender and the receiver, each holding a share of the secret key corresponding to public key $E := \sum \mathbf{C}' \parallel \mathbf{C}'' - \sum \mathbf{C}$, engage in a two-party interactive protocol to compute σ. Actually, this seems to be the path Grin is taking, although, to the best of our knowledge, the problem described above with the original protocol has never been documented.

We show that the spirit of the original *non-interactive* protocol can be salvaged, so a sender can make a payment to a receiver without the latter's active involvement. In our solution the sender first constructs a full-fledged transaction tx spending $\mathbf{C}$ and creating change coins $\mathbf{C}'$ as well as a special output coin $C = \rho H + kG$, and sends tx and the opening (ρ, k) of the special coin to the receiver. (Note that, unlike in the previous case, k is now independent from the keys of the coins in $\mathbf{C}$ and $\mathbf{C}'$.) The receiver then creates a second transaction tx' spending the special coin C and creating its own output coins $\mathbf{C}''$ and aggregates tx and tx'. As intended, this results in a transaction with inputs $\mathbf{C}$ and outputs $\mathbf{C}' \parallel \mathbf{C}''$ since C is removed by cut-through. The only drawback of this procedure is that the final transaction, being the aggregate of two transactions, has two kernel excesses instead of one for the interactive protocol mentioned above.

After specifying our protocol $\mathsf{MW}[\mathsf{Com}, \mathsf{Sig}, \Pi]$, we turn to proving its security in our definitional framework. To this end, we first define two security notions, EUF-NZO and EUF-CRO, tying the commitment scheme and the signature scheme together (cf. Page 65). Assuming that proof system Π is simulation-extractable [DDO+01, Gro06], we show that EUF-NZO-security for the pair $(\mathsf{Com}, \mathsf{Sig})$ implies that MW is resistant to inflation, while EUF-CRO-security implies that MW is resistant to coin theft. Transaction indistinguishability follows from zero knowledge of Π and Com being hiding.

Finally, we consider two natural instantiations of $\mathsf{MW}[\mathsf{Com}, \mathsf{Sig}, \Pi]$. For each, we let Com be the Pedersen commitment scheme [Ped92]. When Sig is instantiated with the Schnorr signature scheme [Sch91], we show that the pair $(\mathsf{Com}, \mathsf{Sig})$ is EUF-NZO- and EUF-CRO-secure under the Discrete Logarithm assumption. When Sig is instantiated with the BLS signature scheme [BLS01], we show that the pair $(\mathsf{Com}, \mathsf{Sig})$ is EUF-NZO- and EUF-CRO-secure under the CDH assumption. Both proofs are in the random-oracle model. BLS signatures have the additional benefit of supporting aggregation [BGLS03, BNN07], so that the ledger kernel always contains a short aggregate signature, independently of the number of transactions that have been added to the ledger. We stress that, unlike Zerocash [BCG+14], none of these two instantiations require a trusted setup.

Future work. As already noted by Poelstra [Poe16], given the aggregate of two transactions for which no cut-through occurred, it is possible to distinguish the inputs and outputs of each original transaction by solving a simple subset sum problem based on the two kernel excesses contained in the aggregate transaction. To achieve indistinguishability of aggregate transactions, Poelstra[10] proposed to add a so-called kernel offset (a random commitment to zero) to each transaction, and to add them when merging transactions (so that any transaction now contains a list of kernel excesses and a single kernel offset). We leave the formal analysis of this proposal, which has already been implemented in Grin, for future work.

5.1 Cryptographic assumptions

Throughout this chapter, we will have to work with lists of commitments and signatures. A list $\mathbf{L} = (x_1, \ldots, x_n)$, also denoted $(x_i)_{i=1}^n$, is a finite sequence. The length of a list $\mathbf{L}$ is denoted $|\mathbf{L}|$.

[10] See https://www.reddit.com/r/Bitcoin/comments/4vub3y/.

For $i = 1, \ldots, |\mathbf{L}|$, the i-th element of $\mathbf{L}$ is denoted $\mathbf{L}[i]$, or L_i when no confusion is possible. By $\mathbf{L}_0 \,\|\, \mathbf{L}_1$ we denote the list $\mathbf{L}_0$ followed by $\mathbf{L}_1$. The empty list is denoted $(\,)$. Given a list $\mathbf{L}$ of elements of an additive group, we let $\sum \mathbf{L}$ denote the sum of all elements of $\mathbf{L}$. Let $\mathbf{L}_0$ and $\mathbf{L}_1$ be two lists, each without repetition. We write $\mathbf{L}_0 \subseteq \mathbf{L}_1$ *iff* each element of $\mathbf{L}_0$ also appears in $\mathbf{L}_1$. We define $\mathbf{L}_0 \cap \mathbf{L}_1$ to be the list of all elements that simultaneously appear in both $\mathbf{L}_0$ and $\mathbf{L}_1$, ordered as in $\mathbf{L}_0$. The difference between $\mathbf{L}_0$ and $\mathbf{L}_1$, denoted $\mathbf{L}_0 - \mathbf{L}_1$, is the list of all elements of $\mathbf{L}_0$ that do not appear in $\mathbf{L}_1$, ordered as in $\mathbf{L}_0$. So, for example $(1,2,3) - (2,4) = (1,3)$. We define the *cut-through* of two lists $\mathbf{L}_0$ and $\mathbf{L}_1$, denoted $\mathsf{cut}(\mathbf{L}_0, \mathbf{L}_1)$, as:

$$\mathsf{cut}(\mathbf{L}_0, \mathbf{L}_1) \coloneqq (\mathbf{L}_0 - \mathbf{L}_1, \mathbf{L}_1 - \mathbf{L}_0) \; .$$

Bilinear groups. Similairly to Chapter 3, we assume the existence of groups $\mathbb{G}, \mathbb{G}_T$ of odd prime order p of length λ and an efficiently computable non-degenerate bilinear map $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. That is, the map e is such that for all $U, V \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p : e(aU, bV) = ab \cdot e(U, V)$, and if U is a generator of $\mathbb{G}$, then $e(U, U)$ is a generator of $\mathbb{G}_T$.

In order to construct an aggregate cash system we will use three cryptographic primitives: a commitment scheme Com (cf. Section 2.3), an aggregate signature scheme Sig, and a non-interactive zero-knowledge proof system Π (cf. Section 2.3). For compatibility reasons, the setup algorithms for each of these schemes are split: a common algorithm $\mathsf{GrGen}(1^\lambda)$ first returns *main* parameters Γ (specifying e.g. an abelian group), and specific algorithms $\mathsf{Com.G}$, $\mathsf{Sig.G}$, and $\Pi.\mathsf{G}$ take as input Γ and return the specific parameters cp, sp, and crs for each primitive. We assume that Γ is contained in cp, sp, and crs.

Recursive aggregate signature scheme. An aggregate signature scheme allows to (publicly) combine an arbitrary number n of signatures (from potentially distinct users and on potentially distinct messages) into a single (ideally short) signature [BGLS03, LMRS04, BNN07]. Traditionally, the syntax of an aggregate signature scheme only allows the aggregation algorithm to take as input individual signatures. We consider aggregate signature schemes supporting *recursive* aggregation, where the aggregation algorithm can take as input aggregate signatures (supported for example by the schemes based on BLS signatures [BGLS03, BNN07]). A recursive aggregate signature scheme Sig consists of the following algorithms:

- $\mathsf{sp} \leftarrow \mathsf{Sig.G}(\Gamma)$: the parameter generation algorithm takes as input main parameters Γ and outputs signature parameters sp, which implicitly define a secret-key space $\mathcal{S}_{\mathsf{sp}}$ and a public-key space $\mathcal{P}_{\mathsf{sp}}$ (we let the message space be $\{0,1\}^*$);

- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Sig.K}(\mathsf{sp})$: the key generation algorithm takes signature parameters sp and outputs a secret key $\mathsf{sk} \in \mathcal{S}_{\mathsf{sp}}$ and a public key $\mathsf{pk} \in \mathcal{P}_{\mathsf{sp}}$;

- $\sigma \leftarrow \mathsf{Sig.S}(\mathsf{sp}, \mathsf{sk}, m)$: the signing algorithm takes as input parameters sp, a secret key $\mathsf{sk} \in \mathcal{S}_{\mathsf{sp}}$, and a message $m \in \{0,1\}^*$ and outputs a signature σ;

- $\sigma \leftarrow \mathsf{Sig.A}(\mathsf{sp}, (\mathbf{L}_0, \sigma_0), (\mathbf{L}_1, \sigma_1))$: the aggregation algorithm takes parameters sp and two pairs of public-key/message lists $\mathbf{L}_i = ((\mathsf{pk}_{i,j}, m_{i,j}))_{j=1}^{|\mathbf{L}_i|}$ and (aggregate) signatures σ_i, $i = 0, 1$; it returns an aggregate signature σ;

- $\mathit{bool} \leftarrow \mathsf{Sig.V}(\mathsf{sp}, \mathbf{L}, \sigma)$: the (deterministic) verification algorithm takes parameters sp, a list $\mathbf{L} = ((\mathsf{pk}_i, m_i))_{i=1}^{|\mathbf{L}|}$ of public-key/message pairs, and an aggregate signature σ; it returns 1 or 0, indicating validity of σ.

Game $\text{EUF-CMA}_{\mathsf{Sig},\mathcal{A}}(\lambda)$

$\mathsf{Q} := (\,);\ \Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$

$\mathsf{sp} \leftarrow \mathsf{Sig.G}(\Gamma)\,;\ (\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{Sig.K}(\mathsf{sp})$

$(\mathbf{L},\sigma) \leftarrow \mathcal{A}^{\text{SIGN}}(\mathsf{pk})$

return $\big(\exists\, m : (\mathsf{pk},m) \in \mathbf{L} \wedge m \notin \mathsf{Q}\big)$ **and** $\mathsf{Sig.V}(\mathsf{sp},\mathbf{L},\sigma)$

Oracle $\text{SIGN}(m)$

$\sigma \leftarrow \mathsf{Sig.S}(\mathsf{sk},m)$

$\mathsf{Q} := \mathsf{Q} \,\|\, (m)$

return σ

Figure 5.1: The EUF-CMA security game for an aggregate signature scheme Sig.

Correctness of a recursive aggregate signature scheme is defined recursively. An aggregate signature scheme is *correct* if for every λ, every message $m \in \{0,1\}^*$, every $\Gamma \in [\mathsf{GrGen}(1^\lambda)]$, $\mathsf{sp} \in [\mathsf{Sig.G}(\Gamma)]$, $(\mathsf{sk},\mathsf{pk}) \in [\mathsf{Sig.K}(\mathsf{sp})]$ and every $(\mathbf{L}_0,\sigma_0),(\mathbf{L}_1,\sigma_1)$ with $\mathsf{Sig.V}(\mathsf{sp},\mathbf{L}_0,\sigma_0) = 1 = \mathsf{Sig.V}(\mathsf{sp},\mathbf{L}_1,\sigma_1)$ we have

$$\Pr\left[\mathsf{Sig.V}\big(\mathsf{sp},((\mathsf{pk},m)),\mathsf{Sig.S}(\mathsf{sp},\mathsf{sk},m)\big) = 1\right] = 1 \ \text{ and}$$
$$\Pr\left[\mathsf{Sig.V}\big(\mathsf{sp},\mathbf{L}_0 \,\|\, \mathbf{L}_1,\mathsf{Sig.A}(\mathsf{sp},(\mathbf{L}_0,\sigma_0),(\mathbf{L}_1,\sigma_1))\big) = 1\right] = 1\ .$$

Note that for any recursive aggregate signature scheme, one can define an aggregation algorithm $\mathsf{Sig.A}'$ that takes as input a list of triples $((\mathsf{pk}_i,m_i,\sigma_i))_{i=1}^n$ and returns an aggregate signature σ for $((\mathsf{pk}_i,m_i))_{i=1}^n$, which is the standard syntax for an aggregate signature scheme. Algorithm $\mathsf{Sig.A}'$ calls Sig.A recursively $n-1$ times, aggregating one signature at a time.

The standard security notion for aggregate signature schemes is *existential unforgeability under chosen-message attack* (EUF-CMA) [BGLS03, BNN07].

Definition 5.1 (EUF-CMA)**.** *Let game* EUF-CMA *be as defined in Fig. 5.1. An aggregate signature scheme* Sig *is* existentially unforgeable under chosen-message attack *if for any* PPT *adversary* $\mathcal{A}$*:*

$$\mathsf{Adv}^{\text{euf-cma}}_{\mathsf{Sig},\mathcal{A}}(\lambda) := \Pr\left[\text{EUF-CMA}_{\mathsf{Sig},\mathcal{A}}(\lambda) = 1\right] = \mathsf{negl}(\lambda)\ .$$

Note that any standard signature scheme can be turned into an aggregate signature scheme by letting the aggregation algorithm simply concatenate signatures, i.e., $\mathsf{Sig.A}(\mathsf{sp},(\mathbf{L}_0,\sigma_0),(\mathbf{L}_1,\sigma_1))$ returns (σ_0,σ_1), but this is not compact. Standard EUF-CMA-security of the original scheme implies EUF-CMA-security in the sense of Definition 5.1 for this construction. This allows us to capture standard and (compact) aggregate signature schemes, such as the ones proposed in [BGLS03, BNN07], in a single framework.

Compatibility. For our aggregate cash system, we require the commitment scheme Com and the aggregate signature scheme Sig to satisfy some "combined" security notions. We say that Com and Sig are *compatible* if they use the same GrGen and if for any λ, any $\Gamma \in [\mathsf{GrGen}(1^\lambda)]$, $\mathsf{cp} \in [\mathsf{Com.G}(\Gamma)]$ and $\mathsf{sp} \in [\mathsf{Sig.G}(\Gamma)]$, the following holds:

- $\mathcal{S}_{\mathsf{sp}} = \mathcal{R}_{\mathsf{cp}}$, i.e., the secret-key space of Sig is the same as the randomness space of Com;
- $\mathcal{P}_{\mathsf{sp}} = \mathcal{C}_{\mathsf{cp}}$, i.e., the public-key space of Sig is the same as the commitment space of Com;
- Sig.K proceeds by drawing $\mathsf{sk} \leftarrow\!\!\$\ \mathcal{R}_{\mathsf{cp}}$ and setting $\mathsf{pk} := \mathsf{Com.C}(\mathsf{cp},0;\mathsf{sk})$.

We define two security notions for compatible commitment and aggregate signature schemes. The first one roughly states that only commitments to zero can serve as signature-verification keys; more precisely, a PPT adversary cannot simultaneously produce a signature for a (set of) freely chosen public key(s) *and* a non-zero opening of (the sum of) the public key(s).

Game EUF-NZO$_{\mathsf{Com},\mathsf{Sig},\mathcal{A}}(\lambda)$

$\Gamma \leftarrow \mathsf{GrGen}(1^\lambda)\,;\ \mathsf{cp} \leftarrow \mathsf{Com.G}(\Gamma)\,;\ \mathsf{sp} \leftarrow \mathsf{Sig.G}(\Gamma)$
$(\mathbf{L}, \sigma, (v, r)) \leftarrow \mathcal{A}(\mathsf{cp}, \mathsf{sp})$
$((X_i, m_i))_{i=1}^n \coloneqq \mathbf{L}$
return $\mathsf{Sig.V}(\mathsf{sp}, \mathbf{L}, \sigma)$ **and** $\sum_{i=1}^n X_i = \mathsf{Com.C}(\mathsf{cp}, v; r)$ **and** $v \neq 0$

Figure 5.2: The EUF-NZO security game for a pair of compatible additively homomorphic commitment and aggregate signature schemes $(\mathsf{Com}, \mathsf{Sig})$.

Definition 5.2 (EUF-NZO)**.** *Let game* EUF-NZO *be as defined in Fig. 5.2. A pair of compatible homomorphic commitment and aggregate signature schemes* $(\mathsf{Com}, \mathsf{Sig})$ *is* existentially unforgeable with non-zero opening *if for any* PPT *adversary* $\mathcal{A}$*:*

$$\mathsf{Adv}^{\text{euf-nzo}}_{\mathsf{Com},\mathsf{Sig},\mathcal{A}}(\lambda) \coloneqq \Pr\left[\text{EUF-NZO}_{\mathsf{Com},\mathsf{Sig},\mathcal{A}}(\lambda) = 1\right] = \mathsf{negl}(\lambda)\ .$$

EUF-NZO-security of the pair $(\mathsf{Com}, \mathsf{Sig})$ implies that Com is binding.

Lemma 5.3. *Let* $(\mathsf{Com}, \mathsf{Sig})$ *be a pair of compatible additively homomorphic commitment and aggregate signature schemes. Then, for any* PPT *adversary* $\mathcal{A}$*, there exists a* PPT *adversary* $\mathcal{B}$ *such that:*

$$\mathsf{Adv}^{\text{bnd}}_{\mathsf{Com},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\text{euf-nzo}}_{\mathsf{Com},\mathsf{Sig},\mathcal{B}}(\lambda)\ .$$

Proof. Let $\mathcal{A}$ be an adversary against the binding security of Com. We construct an adversary $\mathcal{B}$ against the EUF-NZO security of $(\mathsf{Com}, \mathsf{Sig})$. On input $(\mathsf{cp}, \mathsf{sp})$, $\mathcal{B}$ simply runs $\mathcal{A}(\mathsf{cp})$ which returns (v_0, r_0) and (v_1, r_1) such that $v_0 \neq v_1$ and (using the homomorphic property) $\mathsf{Com.C}(\mathsf{cp}, v_0 - v_1; r_0 - r_1) = \mathsf{Com.C}(\mathsf{cp}, 0; 0)$. Then, $\mathcal{B}$ draws $r \leftarrow\!\!\$\, \mathcal{R}_{\mathsf{cp}} = \mathcal{S}_{\mathsf{sp}}$ and computes $C = \mathsf{Com.C}(\mathsf{cp}, 0; r)$. Clearly, $\mathcal{B}$ can produce a signature for public key C for any message since it knows the corresponding secret key r. On the other hand, $C = \mathsf{Com.C}(\mathsf{cp}, v_0 - v_1; r + r_0 - r_1)$, so that $\mathcal{B}$ also has an opening to a non-zero value for C, and hence can win the EUF-NZO game. Since $\mathcal{B}$ is successful exactly when $\mathcal{A}$ is, the result follows. □

The second security definition is more involved. It roughly states that, given a challenge public key C^*, no adversary can produce a signature under $-C^*$. Moreover, we only require the adversary to make a signature under keys $X_1, \ldots, X_n$ of its choice, as long as it knows an opening to the difference between their sum and $-C^*$. This must even hold if the adversary is given a signing oracle for keys related to C^*. Informally, the adversary is faced with the following dilemma: either it picks public keys $X_1, \ldots, X_n$ honestly, so it can produce a signature but it cannot open $\sum X_i + C^*$; or it includes $-C^*$ within the public keys, allowing it to open $\sum X_i + C^*$, but then it cannot produce a signature.

Definition 5.4 (EUF-CRO)**.** *Let game* EUF-CRO *be as defined in Fig. 5.3. A pair of compatible homomorphic commitment and aggregate signature schemes* $(\mathsf{Com}, \mathsf{Sig})$ *is* existentially unforgeable with challenge-related opening *if for any* PPT *adversary* $\mathcal{A}$*:*

$$\mathsf{Adv}^{\text{euf-cro}}_{\mathsf{Com},\mathsf{Sig},\mathcal{A}}(\lambda) \coloneqq \Pr\left[\text{EUF-CRO}_{\mathsf{Com},\mathsf{Sig},\mathcal{A}}(\lambda) = 1\right] = \mathsf{negl}(\lambda)\,.$$

Game EUF-CRO$_{\mathsf{Com},\mathsf{Sig},\mathcal{A}}(\lambda)$

$\Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$; $\mathsf{cp} \leftarrow \mathsf{Com.G}(\Gamma)$
$\mathsf{sp} \leftarrow \mathsf{Sig.G}(\Gamma)$; $(r^*, C^*) \leftarrow \mathsf{Sig.K}(\mathsf{sp})$
$(\mathbf{L}, \sigma, (v, r)) \leftarrow \mathcal{A}^{\textsc{Sign}'}(\mathsf{cp}, \mathsf{sp}, C^*)$; $((X_i, m_i))_{i=1}^n \coloneqq \mathbf{L}$
return $\mathsf{Sig.V}(\mathsf{sp}, \mathbf{L}, \sigma)$ **and** $\sum_{i=1}^n X_i = -C^* + \mathsf{Com.C}(\mathsf{cp}, v; r)$

Oracle $\textsc{Sign}'(a, m)$

$\mathsf{sk}' \coloneqq a + r^*$
return $\mathsf{Sig.S}(\mathsf{sp}, \mathsf{sk}', m)$

Figure 5.3: The EUF-CRO security game for a pair of compatible additively homomorphic commitment and aggregate signature schemes ($\mathsf{Com}, \mathsf{Sig}$).

5.2 Aggregate cash system

Coins. The public parameters pp set up by the cash system specify a coin space $\mathcal{C}_{\mathsf{pp}}$ and a key space $\mathcal{K}_{\mathsf{pp}}$. A *coin* is an element $C \in \mathcal{C}_{\mathsf{pp}}$; to each coin is associated a *coin key* $k \in \mathcal{K}_{\mathsf{pp}}$, which allows spending the coin. The *value* v of a coin is an integer in $[\![0, v_{\max}]\!]$, where $v_{\max}$ is a system parameter. We assume that there exists a function mapping pairs $(v, k) \in [\![0, v_{\max}]\!] \times \mathcal{K}_{\mathsf{pp}}$ to coins in $\mathcal{C}_{\mathsf{pp}}$; we do not assume this mapping to be invertible or even injective.

Ledger. Similarly to any ledger-based currency such as Bitcoin, an aggregate cash system keeps track of available coins in the system via a ledger. We assume the ledger to be unique and available at any time to all users. How users are kept in consensus on the ledger is outside the scope of this manuscript. In our abstraction, a ledger Λ simply provides two attributes: a list of all coins available in the system $\Lambda.\mathsf{out}$, and the total value $\Lambda.\mathsf{sply}$ those coins add up to. We say that a coin C *exists in the ledger* Λ if $C \in \Lambda.\mathsf{out}$.

Transactions. Transactions allow to modify the state of the ledger. Formally, a *transaction* tx provides three attributes: a *coin input list* $\mathsf{tx.in}$, a *coin output list* $\mathsf{tx.out}$, and a *supply* $\mathsf{tx.sply} \in \mathbb{N}$ specifying the amount of money created by tx. We classify transactions into three types. A transaction tx is said to be:

- a *minting transaction* if $\mathsf{tx.sply} > 0$ and $\mathsf{tx.in} = (\,)$; such a transaction creates new coins of total value $\mathsf{tx.sply}$ in the ledger;
- a *transfer transaction* if $\mathsf{tx.sply} = 0$ and $\mathsf{tx.in} \neq (\,)$; such a transaction transfers coins (by spending previous transaction outputs and creating new ones) but does not increase the overall value of coins in the ledger;
- a *mixed transaction* if $\mathsf{tx.sply} > 0$ and $\mathsf{tx.in} \neq (\,)$.

Pre-transactions. Pre-transactions allow users to transfer money to each other. Formally, a *pre-transaction* provides three attributes: a *coin input list* $\mathsf{ptx.in}$, a *list of change coins* $\mathsf{ptx.chg}$, and a *remainder* $\mathsf{ptx.rmdr}$. When Alice wants to send money worth ρ to Bob, she selects coins of hers of total value $v \geq \rho$ and specifies the desired values for her change coins when $v > \rho$. The resulting pre-transaction ptx has therefore some input coin list $\mathsf{ptx.in}$ with total amount v, a change coin list $\mathsf{ptx.chg}$, and some remainder $\rho = \mathsf{ptx.rmdr}$. Alice sends this pre-transaction (via a secure channel) to Bob, who, in turn, finalizes it into a valid transaction and adds it to the ledger.

Aggregate cash system. An aggregate cash system Cash consists of the following algorithms:

- $(\mathsf{pp}, \Lambda) \leftarrow \mathsf{Cash.G}(1^\lambda, v_{\max})$: the setup algorithm takes as input the security parameter λ in unary and a maximal coin value $v_{\max}$ and returns public parameters pp and an initial (empty) ledger Λ.
- $(\mathsf{tx}, \mathbf{k}) \leftarrow \mathsf{Cash.M}(\mathsf{pp}, \mathbf{v})$: the mint algorithm takes as input a list of values $\mathbf{v}$ and returns a minting transaction tx and a list of coin keys $\mathbf{k}$ for the coins in $\mathsf{tx.out}$, such that the supply of tx is the sum of the values $\mathbf{v}$.
- $(\mathsf{ptx}, \mathbf{k}') \leftarrow \mathsf{Cash.S}(\mathsf{pp}, (\mathbf{C}, \mathbf{v}, \mathbf{k}), \mathbf{v}')$: the sending algorithm takes as input a list of coins $\mathbf{C}$ together with the associated lists of values $\mathbf{v}$ and secret keys $\mathbf{k}$ and a list of *change values* $\mathbf{v}'$ whose sum is at most the sum of the input values $\mathbf{v}$; it returns a pre-transaction ptx and a list of keys $\mathbf{k}'$ for the change coins of ptx, such that the remainder of ptx is the sum of the values $\mathbf{v}$ minus the sum of the values $\mathbf{v}'$.
- $(\mathsf{tx}, \mathbf{k}'') \leftarrow \mathsf{Cash.R}(\mathsf{pp}, \mathsf{ptx}, \mathbf{v}'')$: the receiving algorithm takes as input a pre-transaction ptx and a list of values $\mathbf{v}''$ whose sum equals the remainder of ptx; it returns a transfer transaction tx and a list of secret keys $\mathbf{k}''$ for the fresh coins in the output of tx, one for each value in $\mathbf{v}''$.
- $\Lambda' \leftarrow \mathsf{Cash.L}(\mathsf{pp}, \Lambda, \mathsf{tx})$: the ledger algorithm takes as input the ledger Λ and a transaction tx to be included in Λ; it returns an updated ledger Λ' or $\perp$.
- $\mathsf{tx} \leftarrow \mathsf{Cash.A}(\mathsf{pp}, \mathsf{tx}_0, \mathsf{tx}_1)$: the transaction aggregation algorithm takes as input two transactions tx_0 and tx_1 whose input coin lists are disjoint and whose output coin lists are disjoint; it returns a transaction tx whose supply is the sum of the supplies of tx_0 and tx_1 and whose input and output coin list is the cut-through of $\mathsf{tx}_0.\mathsf{in} \,\|\, \mathsf{tx}_1.\mathsf{in}$ and $\mathsf{tx}_0.\mathsf{out} \,\|\, \mathsf{tx}_1.\mathsf{out}$.

We say that an aggregate cash system Cash is correct if its procedures $\mathsf{Cash.G}$, $\mathsf{Cash.M}$, $\mathsf{Cash.S}$, $\mathsf{Cash.R}$, $\mathsf{Cash.L}$, and $\mathsf{Cash.A}$ behave as expected with overwhelming probability (that is, we allow that with negligible probability things can go wrong, typically, because an algorithm could generate the same coin twice). We give a formal definition that uses two auxiliary procedures: Cons, which checks if a list of coins $\mathbf{C}$ is consistent with respect to values $\mathbf{v}$ and keys $\mathbf{k}$; and V, which given as input a ledger or a (pre-)transaction determines if they respect some notion of cryptographic validity.

Definition 5.5 (Correctness)**.** *An aggregate cash system* Cash *is* correct *if there exist procedures* $\mathsf{V}(\cdot,\cdot)$ *and* $\mathsf{Cons}(\cdot,\cdot,\cdot,\cdot)$ *such that for any* $v_{\max} \in \mathbb{N}$ *and (not necessarily* PPT*)* $\mathcal{A}_\mathsf{M}, \mathcal{A}_\mathsf{S}, \mathcal{A}_\mathsf{R}, \mathcal{A}_\mathsf{A}$ *and* $\mathcal{A}_\mathsf{L}$ *the following functions are overwhelming in* λ*:* $\Pr\left[(\mathsf{pp}, \Lambda) \leftarrow \mathsf{Cash.G}(1^\lambda, v_{\max}) \ : \ \mathsf{Cash.V}(\mathsf{pp}, \Lambda)\right]$

$$\Pr\left[\ (\mathsf{pp}, \Lambda) \leftarrow \mathsf{Cash.G}(1^\lambda, v_{\max}) \ : \ \mathsf{Cash.V}(\mathsf{pp}, \Lambda)\ \right]$$

$$\Pr\left[\begin{array}{l} (\mathsf{pp}, \Lambda) \leftarrow \mathsf{Cash.G}(1^\lambda, v_{\max}) \\ \mathbf{v} \leftarrow \mathcal{A}_\mathsf{M}(\mathsf{pp}, \Lambda) \\ (\mathsf{tx}, \mathbf{k}) \leftarrow \mathsf{Cash.M}(\mathsf{pp}, \mathbf{v}) \end{array} : \mathbf{v} \in [\![0, v_{\max}]\!]^* \ \Rightarrow \left(\begin{array}{l} \mathsf{Cash.V}(\mathsf{pp}, \mathsf{tx}) \ \wedge \ \mathsf{tx.in} = (\,) \wedge \\ \mathsf{tx.sply} = \sum \mathbf{v} \ \wedge \\ \mathsf{Cons}(\mathsf{pp}, \mathsf{tx.out}, \mathbf{v}, \mathbf{k}) \end{array}\right)\right]$$

$$\Pr\left[\begin{array}{l} (\mathsf{pp}, \Lambda) \leftarrow \mathsf{Cash.G}(1^\lambda, v_{\max}) \\ (\mathbf{C}, \mathbf{v}, \mathbf{k}, \mathbf{v}') \leftarrow \mathcal{A}_\mathsf{S}(\mathsf{pp}, \Lambda) \\ (\mathsf{ptx}, \mathbf{k}') \leftarrow \mathsf{Cash.S}(\mathsf{pp}, (\mathbf{C}, \mathbf{v}, \mathbf{k}), \mathbf{v}') \end{array} : \begin{array}{l} \left(\begin{array}{l} \mathsf{Cons}(\mathsf{pp}, \mathbf{C}, \mathbf{v}, \mathbf{k}) \ \wedge \ \mathbf{v} \,\|\, \mathbf{v}' \in [\![0, v_{\max}]\!]^* \ \wedge \\ \sum \mathbf{v} - \sum \mathbf{v}' \in [\![0, v_{\max}]\!] \end{array}\right) \\ \Rightarrow \left(\begin{array}{l} \mathsf{Cash.V}(\mathsf{pp}, \mathsf{ptx}) \ \wedge \ \mathsf{ptx.in} = \mathbf{C} \ \wedge \\ \mathsf{ptx.rmdr} = \sum \mathbf{v} - \sum \mathbf{v}' \ \wedge \\ \mathsf{Cons}(\mathsf{pp}, \mathsf{ptx.chg}, \mathbf{v}', \mathbf{k}') \end{array}\right) \end{array}\right]$$

$$\Pr\left[\begin{array}{l}(\mathsf{pp},\Lambda)\leftarrow\mathsf{Cash.G}(1^\lambda,v_{\max})\\(\mathsf{ptx},\mathbf{v}'')\leftarrow\mathcal{A}_{\mathsf{R}}(\mathsf{pp},\Lambda)\\(\mathsf{tx},\mathbf{k}'')\leftarrow\mathsf{Cash.R}(\mathsf{pp},\mathsf{ptx},\mathbf{v}'')\end{array}:\begin{array}{l}\Big(\ \mathsf{Cash.V}(\mathsf{pp},\mathsf{ptx})\wedge\mathbf{v}''\in[\![0,v_{\max}]\!]^*\wedge\mathsf{ptx.rmdr}=\sum\mathbf{v}''\ \Big)\\ \Rightarrow\left(\begin{array}{l}\mathsf{Cash.V}(\mathsf{pp},\mathsf{tx})\ \wedge\ \mathsf{tx.sply}=0\ \wedge\\ \mathsf{tx.in}=\mathsf{ptx.in}\ \wedge\ \mathsf{ptx.chg}\subseteq\mathsf{tx.out}\ \wedge\\ \mathsf{Cons}(\mathsf{pp},\mathsf{tx.out}-\mathsf{ptx.chg},\mathbf{v}'',\mathbf{k}'')\end{array}\right)\end{array}\right]$$

$$\Pr\left[\begin{array}{l}(\mathsf{pp},\Lambda)\leftarrow\mathsf{Cash.G}(1^\lambda,v_{\max})\\(\mathsf{tx}_0,\mathsf{tx}_1)\leftarrow\mathcal{A}_{\mathsf{A}}(\mathsf{pp},\Lambda)\\\mathsf{tx}\leftarrow\mathsf{Cash.A}(\mathsf{pp},\mathsf{tx}_0,\mathsf{tx}_1)\end{array}:\begin{array}{l}\left(\begin{array}{l}\mathsf{Cash.V}(\mathsf{pp},\mathsf{tx}_0)\ \wedge\ \mathsf{tx}_0.\mathsf{in}\cap\mathsf{tx}_1.\mathsf{in}=(\,)\ \wedge\\ \mathsf{Cash.V}(\mathsf{pp},\mathsf{tx}_1)\ \wedge\ \mathsf{tx}_0.\mathsf{out}\cap\mathsf{tx}_1.\mathsf{out}=(\,)\end{array}\right)\\ \Rightarrow\left(\begin{array}{l}\mathsf{Cash.V}(\mathsf{pp},\mathsf{tx})\ \wedge\ \mathsf{tx.sply}=\mathsf{tx}_0.\mathsf{sply}+\mathsf{tx}_1.\mathsf{sply}\ \wedge\\ \mathsf{tx.in}=(\mathsf{tx}_0.\mathsf{in}\,\|\,\mathsf{tx}_1.\mathsf{in})-(\mathsf{tx}_0.\mathsf{out}\,\|\,\mathsf{tx}_1.\mathsf{out})\ \wedge\\ \mathsf{tx.out}=(\mathsf{tx}_0.\mathsf{out}\,\|\,\mathsf{tx}_1.\mathsf{out})-(\mathsf{tx}_0.\mathsf{in}\,\|\,\mathsf{tx}_1.\mathsf{in})\end{array}\right)\end{array}\right]$$

$$\Pr\left[\begin{array}{l}(\mathsf{pp},\Lambda)\leftarrow\mathsf{Cash.G}(1^\lambda,v_{\max})\\(\Lambda,\mathsf{tx})\leftarrow\mathcal{A}_{\mathsf{L}}(\mathsf{pp},\Lambda)\\\Lambda'\leftarrow\mathsf{Cash.L}(\mathsf{pp},\Lambda,\mathsf{tx})\end{array}:\begin{array}{l}\left(\begin{array}{l}\mathsf{Cash.V}(\mathsf{pp},\Lambda)\ \wedge\ \mathsf{Cash.V}(\mathsf{pp},\mathsf{tx})\ \wedge\\ \mathsf{tx.in}\subseteq\Lambda.\mathsf{out}\ \wedge\ \mathsf{tx.out}\cap\Lambda.\mathsf{out}=(\,)\end{array}\right)\\ \Rightarrow\left(\begin{array}{l}\Lambda'\neq\perp\ \wedge\ \mathsf{Cash.V}(\mathsf{pp},\Lambda')\ \wedge\\ \Lambda'.\mathsf{out}=(\Lambda.\mathsf{out}-\mathsf{tx.in})\,\|\,\mathsf{tx.out}\ \wedge\\ \Lambda'.\mathsf{sply}=\Lambda.\mathsf{sply}+\mathsf{tx.sply}\end{array}\right)\end{array}\right]$$

Security definitions

Security against inflation. A sound payment system must ensure that the only way money can be created is via the supply of transactions, typically minting transactions. This means that for any tx the total value of the output coins should be equal to the sum of the total value of the input coins plus the supply $\mathsf{tx.sply}$ of the transaction. Since coin values are not deducible from a transaction (this is one of the privacy features of such a system), we define the property at the level of the ledger Λ.

We say that a cash system is resistant to inflation if no adversary can spend coins from $\Lambda.\mathsf{out}$ worth more than $\Lambda.\mathsf{sply}$. The adversary's task is thus to create a pre-transaction whose remainder is strictly greater than $\Lambda.\mathsf{sply}$; validity of the pre-transaction is checked by completing it to a transaction via R and adding it to the ledger via L. This is captured by the definition below.

Definition 5.6 (Inflation-resistance)**.** *We say that an aggregate cash system* Cash *is secure against inflation if for any* $v_{\max}$ *and any* PPT *adversary* $\mathcal{A}$*:*

$$\mathsf{Adv}^{\mathrm{infl}}_{\mathsf{Cash},\mathcal{A}}(\lambda,v_{\max}) \coloneqq \Pr\left[\mathrm{INFL}_{\mathsf{Cash},\mathcal{A}}(\lambda,v_{\max})=1\right]=\mathsf{negl}(\lambda)\,,$$

where $\mathrm{INFL}_{\mathsf{Cash},\mathcal{A}}(\lambda,v_{\max})$ *is defined in Fig. 5.4.*

Security against coin theft. Besides inflation, which protects the soundness of the system as a whole, the second security notion protects individual users. It requires that only a user can spend coins belonging to him, where ownership of a coin amounts to knowledge of the coin secret key. This is formalized by the experiment in Fig. 5.5, which proceeds as follows. The challenger sets up the system and maintains the ledger Λ throughout the game (we assume that the consensus protocol provides this). The adversary can add any valid transaction to the ledger through an oracle LEDGER.

Game $\mathrm{INFL}_{\mathsf{Cash},\mathcal{A}}(\lambda, v_{\max})$

$(\mathsf{pp}, \Lambda_0) \leftarrow \mathsf{Cash.G}(1^\lambda, v_{\max})$
$(\Lambda, \mathsf{ptx}, \mathbf{v}) \leftarrow \mathcal{A}(\mathsf{pp}, \Lambda_0)$
$(\mathsf{tx}, \mathbf{k}) \leftarrow \mathsf{Cash.R}(\mathsf{pp}, \mathsf{ptx}, \mathbf{v})$
return $\bot \not\leftarrow \mathsf{Cash.L}(\mathsf{pp}, \Lambda, \mathsf{tx})$ **and** $\Lambda.\mathsf{sply} < \sum \mathbf{v}$

Figure 5.4: Game formalizing resistance to inflation of a cash system Cash.

The challenger also simulates an honest user and manages her coins; in particular, it maintains a list Hon, which represents the coins that the honest user expects to own in the ledger. The game also maintains two hash tables Val and Key that map coins produced by the game to their values and keys. We write e.g. $\mathsf{Val}(C) := v$ to mean that the pair (C, v) is added to Val and let $\mathsf{Val}(C)$ denote the value v for which (C, v) is in Val. This naturally generalizes to lists letting $\mathsf{Val}(\mathbf{C})$ be the list $\mathbf{v}$ such that (C_i, v_i) is in Val for all i.

The adversary can interact with the honest user and the ledger using the following oracles:

- Mint is an oracle that mints coins for the honest user. It takes as input a vector of values $\mathbf{v}$, creates a minting transaction tx together with the secret keys of the output coins, adds tx to the ledger and appends the newly created coins to Hon.

- Receive lets the adversary send coins to the honest user. The oracle takes as input a pre-transaction ptx and output values $\mathbf{v}$; it completes ptx to a transaction tx creating output coins with values $\mathbf{v}$, adds tx to the ledger, and appends the newly created coins to Hon.

- Send lets the adversary make an honest user send coins to it. It takes as input a list $\mathbf{C}$ of coins contained in Hon and a list of change values $\mathbf{v}'$; it also checks that none of the coins in $\mathbf{C}$ has been queried to Send before (an honest user does not double-spend). It returns a pre-transaction ptx spending the coins from $\mathbf{C}$ and creating change output coins with values $\mathbf{v}'$. The oracle only produces a pre-transaction and returns it to the adversary, but it does not alter the ledger. This is why the list Hon of honest coins is not altered either; in particular, the sent coins $\mathbf{C}$ still remain in Hon.

- Ledger lets the adversary commit a transaction tx to the ledger. If the transaction output contains the (complete) set of change coins of a pre-transaction ptx previously sent to the adversary, then these change coins are added to Hon, while the input coins of ptx are removed from Hon.

Note that the list Hon represents the coins that the honest user should consider hers, given the system changes induced by the oracle calls: coins received directly from the adversary via Receive or as fresh coins via Mint are added to Hon. Coins sent to the adversary in a pre-transaction ptx via Send are only removed *once all change coins of* ptx *have been added* to the ledger via Ledger. Note also that, given these oracles, the adversary can simulate transfers between honest users. It can simply call Send to receive an honest pre-transaction ptx and then call Receive to have the honest user receive ptx.

The winning condition of the game is now simply that Hon does not reflect what the honest user would expect, namely Hon is not fully contained in the ledger (because the adversary managed to spend a coin that is still in Hon, which amounts to stealing it from the honest user).

Game $\text{STEAL}_{\mathsf{Cash},\mathcal{A}}(\lambda, v_{\max})$

$(\mathsf{pp}, \Lambda) \leftarrow \mathsf{Cash.G}(1^\lambda, v_{\max})$
$\mathsf{Hon}, \mathsf{Val}, \mathsf{Key}, \mathsf{Ptx} := (\,)$
$\mathcal{A}^{\text{MINT},\text{SEND},\text{RECEIVE},\text{LEDGER}}(\mathsf{pp}, \Lambda)$
return $(\mathsf{Hon} \not\subseteq \Lambda.\mathsf{out})$

Aux function $\mathsf{Store}(\mathbf{C}, \mathbf{v}, \mathbf{k})$

$\mathsf{Val}(\mathbf{C}) := \mathbf{v}\,;\ \mathsf{Key}(\mathbf{C}) := \mathbf{k}$

Oracle $\text{MINT}(\mathbf{v})$

$(\mathsf{tx}, \mathbf{k}) \leftarrow \mathsf{Cash.M}(\mathsf{pp}, \mathbf{v})$
$\Lambda \leftarrow \mathsf{Cash.L}(\mathsf{pp}, \Lambda, \mathsf{tx})$
$\mathsf{Hon} := \mathsf{Hon} \,\|\, \mathsf{tx.out}$
$\mathsf{Store}(\mathsf{tx.out}, \mathbf{v}, \mathbf{k})$
return tx

Oracle $\text{SEND}(\mathbf{C}, \mathbf{v}')$

if $\mathbf{C} \not\subseteq \mathsf{Hon}$ **or** $\bigcup_{\mathsf{ptx}\in\mathsf{Ptx}} \mathsf{ptx.in} \cap \mathbf{C} \neq (\,)$
  return $\perp$ // only honest coins never sent can be queried
$(\mathsf{ptx}, \mathbf{k}') \leftarrow \mathsf{Cash.S}\big(\mathsf{pp}, (\mathbf{C}, \mathsf{Val}(\mathbf{C}), \mathsf{Key}(\mathbf{C})), \mathbf{v}'\big)$
$\mathsf{Store}(\mathsf{ptx.chg}, \mathbf{v}', \mathbf{k}')\,;\ \mathsf{Ptx} := \mathsf{Ptx} \,\|\, (\mathsf{ptx})$
return ptx

Oracle $\text{RECEIVE}(\mathsf{ptx}, \mathbf{v})$

$(\mathsf{tx}, \mathbf{k}) \leftarrow \mathsf{Cash.R}(\mathsf{pp}, \mathsf{ptx}, \mathbf{v})$
$\Lambda' \leftarrow \text{LEDGER}(\mathsf{tx})$ // updates Hon
if $\Lambda' = \perp$: **return** $\perp$
$\mathsf{Hon} := \mathsf{Hon} \,\|\, (\mathsf{tx.out} - \mathsf{ptx.chg})$
$\mathsf{Store}(\mathsf{tx.out} - \mathsf{ptx.chg}, \mathbf{v}, \mathbf{k})$
return tx

Oracle $\text{LEDGER}(\mathsf{tx})$

$\Lambda' \leftarrow \mathsf{Cash.L}(\mathsf{pp}, \Lambda, \mathsf{tx})$
if $\Lambda' = \perp$: **return** $\perp$ **else** : $\Lambda := \Lambda'$
for all $\mathsf{ptx} \in \mathsf{Ptx}$ **do**
  if $\mathsf{ptx.chg} \subseteq \mathsf{tx.out}$
  // if all change of ptx now in ledger
    $\mathsf{Ptx} := \mathsf{Ptx} - (\mathsf{ptx})$
    $\mathsf{Hon} := (\mathsf{Hon} - \mathsf{ptx.in}) \,\|\, \mathsf{ptx.chg}$
    // consider input of ptx consumed
return Λ

Figure 5.5: Game formalizing resistance to coin theft of a cash system Cash.

Definition 5.7 (Theft-resistance)**.** *We say that an aggregate cash system* Cash *is* secure against coin theft *if for any* $v_{\max}$ *and any* PPT *adversary* $\mathcal{A}$*:*

$$\mathsf{Adv}^{\text{steal}}_{\mathsf{Cash},\mathcal{A}}(\lambda, v_{\max}) := \Pr\left[\text{STEAL}_{\mathsf{Cash},\mathcal{A}}(\lambda, v_{\max}) = 1\right] = \mathsf{negl}(\lambda)\,,$$

where $\text{STEAL}_{\mathsf{Cash},\mathcal{A}}(\lambda, v_{\max})$ *is defined in Fig. 5.5.*

Transaction indistinguishability. An important security feature that Mimblewimble inherits from Confidential Transactions [Max15] is that the amounts involved in a transaction are hidden so that only the sender and the receiver know how much money is involved. In addition, a transaction completely hides which inputs paid which outputs and which coins were change and which were added by the receiver.

We formalize this via the following game, specified in Fig. 5.6. The adversary submits two sets of values $(\mathbf{v}_0, \mathbf{v}'_0, \mathbf{v}''_0)$ and $(\mathbf{v}_1, \mathbf{v}'_1, \mathbf{v}''_1)$ representing possibles values for input coins, change coins and receiver's coins of a transaction. The game creates a transaction with values either from the first or the second set and the adversary must guess which. For the transaction to be valid, we must have $\sum \mathbf{v}_b = \sum \mathbf{v}'_b + \sum \mathbf{v}''_b$ for both $b = 0, 1$. Moreover, transactions do not hide the *number* of input and output coins. We therefore also require that $|\mathbf{v}_0| = |\mathbf{v}_1|$ and $|\mathbf{v}'_0| + |\mathbf{v}''_0| = |\mathbf{v}'_1| + |\mathbf{v}''_1|$ (note that e.g. the number of change coins can differ).

Game IND-TX$^b_{\mathsf{Cash},\mathcal{A}}(\lambda, v_{\max})$

$(\mathsf{pp}, \Lambda) \leftarrow \mathsf{Cash.G}(1^\lambda, v_{\max})$
$b' \leftarrow \mathcal{A}^{\mathrm{Tx}}(\mathsf{pp}, \Lambda)$
return b'

Oracle $\mathrm{Tx}((\mathbf{v}_0, \mathbf{v}'_0, \mathbf{v}''_0), (\mathbf{v}_1, \mathbf{v}'_1, \mathbf{v}''_1))$

if not $(\mathbf{v}_0, \mathbf{v}'_0, \mathbf{v}''_0, \mathbf{v}_1, \mathbf{v}'_1, \mathbf{v}''_1 \in [\![0, v_{\max}]\!]^*)$
  return $\perp$
if $|\mathbf{v}_0| \neq |\mathbf{v}_1|$ **or** $|\mathbf{v}'_0| + |\mathbf{v}''_0| \neq |\mathbf{v}'_1| + |\mathbf{v}''_1|$
  return $\perp$ // as number of coins is not hidden
if $\sum \mathbf{v}_0 \neq \sum(\mathbf{v}'_0 \,\|\, \mathbf{v}''_0)$ **or** $\sum \mathbf{v}_1 \neq \sum(\mathbf{v}'_1 \,\|\, \mathbf{v}''_1)$
  return $\perp$ // as transactions must be balanced
$(\mathsf{tx}, \mathbf{k}) \leftarrow \mathsf{M}(\mathsf{pp}, \mathbf{v}_b)$
$(\mathsf{ptx}, \mathbf{k}') \leftarrow \mathsf{S}(\mathsf{pp}, (\mathsf{tx.out}, \mathbf{v}_b, \mathbf{k}), \mathbf{v}'_b)$
$(\mathsf{tx}^*, \mathbf{k}'') \leftarrow \mathsf{R}(\mathsf{pp}, \mathsf{ptx}, \mathbf{v}''_b)$
return tx^*

Figure 5.6: Game formalizing transaction indistinguishability of a cash system Cash.

Definition 5.8 (Transaction indistinguishability)**.** *We say that an aggregate cash system* Cash *is* transaction-indistinguishable *if for any* $v_{\max}$ *and any* PPT *adversary* $\mathcal{A}$*:*

$$\mathsf{Adv}^{\text{tx-ind}}_{\mathsf{Cash},\mathcal{A}}(\lambda, v_{\max}) := \left|\Pr\left[\text{TX-IND}^0_{\mathsf{Cash},\mathcal{A}}(\lambda, v_{\max}) = 1\right] - \Pr\left[\text{TX-IND}^1_{\mathsf{Cash},\mathcal{A}}(\lambda, v_{\max}) = 1\right]\right|$$

is negligible, where TX-IND$^b_{\mathsf{Cash},\mathcal{A}}(\lambda, v_{\max})$ *is defined in Fig. 5.6.*

5.3 Construction of an aggregate cash system

Description

Let Com be a homomorphic commitment scheme such that for $\mathsf{cp} \leftarrow \mathsf{Com.G}(\mathsf{GrGen}(1^\lambda))$ we have value space $\mathcal{V}_{\mathsf{cp}} = \mathbb{Z}_p$ with p of length λ (such as the Pedersen scheme). Let Sig be an aggregate signature scheme that is compatible with Com. For $v_{\max} \in \mathbb{N}$, let $\mathsf{R}_{v_{\max}}$ be the (efficiently computable) relation on commitments with values at most $v_{\max}$, i.e.:

$$\mathsf{R}_{v_{\max}} := \{(\mathsf{cp}, C), (v, r)) \mid \ \wedge\ C = \mathsf{Com.C}(\mathsf{cp}, v; r) \ \wedge\ v \in [\![0, v_{\max}]\!]\} \ ,$$

where we implicitly assume that the family depends also on $\Gamma \in [\mathsf{GrGen}(1^\lambda)]$, and that it is fixed for all statements. Let Π be a simulation-extractable NIZK proof system for the family of relations $\mathsf{R} = \{\mathsf{R}_{v_{\max}}\}_{v_{\max}}$.

For notational simplicity, we will use the following vectorial notation for Com, R, and Π: given $\mathbf{C}$, $\mathbf{v}$, and $\mathbf{r}$ with $|\mathbf{C}| = |\mathbf{v}| = |\mathbf{r}|$, we let

$$\begin{aligned}
\mathsf{Com.C}(\mathsf{cp}, \mathbf{v}; \mathbf{r}) &:= (\mathsf{Com.C}(\mathsf{cp}, v_i; r_i))_{i=1}^{|\mathbf{v}|} \ , \\
\mathsf{R}_{v_{\max}}((\mathsf{cp}, \mathbf{C}), (\mathbf{v}, \mathbf{r})) &:= \textstyle\bigwedge_{i=1}^{|\mathbf{C}|} \mathsf{R}_{v_{\max}}(\Gamma_{\mathsf{cp}}, (\mathsf{cp}, C_i), (v_i, r_i)) \ , \\
\Pi.\mathsf{P}(\mathsf{crs}, (\mathsf{cp}, \mathbf{C}), (\mathbf{v}, \mathbf{r})) &:= (\Pi.\mathsf{P}(\mathsf{crs}, (\mathsf{cp}, C_i), (v_i, r_i)))_{i=1}^{|\mathbf{C}|} \ , \\
\Pi.\mathsf{V}(\mathsf{crs}, (\mathsf{cp}, \mathbf{C}), \pi) &:= \textstyle\bigwedge_{i=1}^{|\mathbf{C}|} \Pi.\mathsf{V}(\mathsf{crs}, (\mathsf{cp}, C_i), \pi_i) \ ,
\end{aligned}$$

and likewise for Π.Sim. We also assume that messages are the empty string $\mathbf{e}$ if they are omitted from Sig.V and Sig.A; that is, we overload notation and let:

$$\mathsf{Sig.V}(\mathsf{sp}, (X_i)_{i=1}^n, \sigma) := \mathsf{Sig.V}(\mathsf{sp}, ((X_i, \mathbf{e}))_{i=1}^n, \sigma)$$

$\underline{\mathsf{MW.Coin}((\mathsf{cp},\mathsf{sp},\mathsf{crs}),\mathbf{v})}$

$(\mathbf{C},\mathbf{k}) \coloneqq \mathsf{Com.C}(\mathsf{cp},\mathbf{v})$
$\pi \leftarrow \Pi.\mathsf{P}(\mathsf{crs},(\mathsf{cp},\mathbf{C}),(\mathbf{v},\mathbf{k}))$
return $(\mathbf{C},\mathbf{v},\pi)$

$\underline{\mathsf{MW.MkTx}((\mathsf{cp},\mathsf{sp},\mathsf{crs}),(\mathbf{C},\mathbf{v},\mathbf{k}),\hat{\mathbf{v}})}$

if $\neg\,\mathsf{Cons}(\mathsf{pp},\mathbf{C},\mathbf{v},\mathbf{k})$: **return** $\perp$
$s \coloneqq \sum\hat{\mathbf{v}} - \sum\mathbf{v}$
if $\mathbf{v} \,\|\, \hat{\mathbf{v}} \not\subseteq [\![0, v_{\max}]\!]^*$ **or** $s < 0$
  return $\perp$
$(\hat{\mathbf{C}},\hat{\mathbf{k}},\hat{\pi}) \leftarrow \mathsf{MW.Coin}(\mathsf{pp},\hat{\mathbf{v}})$
$E \coloneqq \sum\hat{\mathbf{C}} - \sum\mathbf{C} - \mathsf{Com.C}(\mathsf{cp},s;0)$
$\sigma \leftarrow \mathsf{Sig.S}(\mathsf{sp},\sum\hat{\mathbf{k}} - \sum\mathbf{k},\mathbf{e})$
$K \coloneqq (\hat{\pi},E,\sigma)$
$\mathsf{tx} \coloneqq (s,\mathbf{C},\hat{\mathbf{C}},K)$
return $(\mathsf{tx},\hat{\mathbf{k}})$

$\underline{\mathsf{MW.Cons}((\mathsf{cp},\mathsf{sp},\mathsf{crs}),\mathbf{C},\mathbf{v},\mathbf{k})}$

return $|\mathbf{C}| = |\mathbf{v}| = |\mathbf{k}|$ **and** $\mathbf{v} \in [\![0, v_{\max}]\!]^*$
  and $(\forall\, i \neq j : C_i \neq C_j)$
  and $\mathbf{C} = \mathsf{Com.C}(\mathsf{cp},\mathbf{v};\mathbf{k})$
// $\mathsf{Cons}(\mathsf{pp},(\,),(\,),(\,))$ returns 1

$\underline{\mathsf{MW.V}((\mathsf{cp},\mathsf{sp},\mathsf{crs}),\mathsf{tx})}$

if $\mathsf{tx} = (0,(\,),(\,),((\,),(\,),\mathbf{e}))$: **return** 1
$(s,\mathbf{C},\hat{\mathbf{C}},K) \coloneqq \mathsf{tx}$; $(\pi,\mathbf{E},\sigma) \coloneqq K$
$\mathsf{Exc} \coloneqq \sum\hat{\mathbf{C}} - \sum\mathbf{C} - \mathsf{Com.C}(\mathsf{cp},s;0)$
return $(\forall\, i \neq j : C_i \neq C_j \wedge \hat{C}_i \neq \hat{C}_j)$ **and** $\mathbf{C} \cap \hat{\mathbf{C}} = (\,)$
  and $s \geq 0$ **and** $\Pi.\mathsf{V}(\mathsf{crs},\hat{\mathbf{C}},\pi)$ **and**
  $\sum\mathbf{E} = \mathsf{Exc}$ **and** $\mathsf{Sig.V}(\mathsf{sp},\mathbf{E},\sigma)$

$\underline{\mathsf{MW.V}(\mathsf{pp},\mathsf{ptx})}$

$(\mathsf{tx},\rho,k') \coloneqq \mathsf{ptx}$
return $\mathsf{MW.V}(\mathsf{pp},\mathsf{tx})$ **and** $\mathsf{tx.sply} = 0$ **and**
  $\mathsf{MW.Cons}(\mathsf{pp},\mathsf{tx.out}[|\mathsf{tx.out}|],\rho,k')$

$\underline{\mathsf{MW.V}(\mathsf{pp},\Lambda)}$

$\mathsf{tx} \coloneqq \Lambda$ // interpret Λ as transaction
return $\mathsf{MW.V}(\mathsf{pp},\mathsf{tx})$ **and** $\mathsf{tx.in} = (\,)$

Figure 5.7: Auxiliary algorithms for the MW aggregate cash system.

and likewise for $\mathsf{Sig.A}(\mathsf{sp},((X_{0,i})_{i=1}^{n_0},\sigma_0),((X_{1,i})_{i=1}^{n_1},\sigma_1))$.

From Com, Sig and Π we construct an aggregate cash system MW[Com, Sig, Π] as follows. The public parameters pp consist of commitment and signature parameters cp, sp, and a CRS for Π. A *coin key* $k \in \mathcal{K}_{\mathsf{pp}}$ is an element of the randomness space $\mathcal{R}_{\mathsf{cp}}$ of the commitment scheme, i.e., $\mathcal{K}_{\mathsf{pp}} = \mathcal{R}_{\mathsf{cp}}$. A *coin* $C = \mathsf{Com.C}(\mathsf{cp},v;k)$ is a commitment to the value v of the coin using the coin key k as randomness. Hence, $\mathcal{C}_{\mathsf{pp}} = \mathcal{C}_{\mathsf{cp}}$.

A transaction $\mathsf{tx} = (s,\mathbf{C},\hat{\mathbf{C}},K)$ consists of a supply $\mathsf{tx.sply} = s$, an input coin list $\mathsf{tx.in} = \mathbf{C}$, an output coin list $\mathsf{tx.out} = \hat{\mathbf{C}}$, and a kernel K. The kernel K is a triple $(\pi,\mathbf{E},\sigma)$ where π is a list of range proofs for the output coins, $\mathbf{E}$ is a non-empty list of signature-verification keys (which are of the same form as commitments) called *kernel excesses*, and σ is an (aggregate) signature. We define the *excess of the transaction* tx, denoted $\mathsf{Exc}(\mathsf{tx})$, as the sum of outputs minus the sum of inputs, with the supply s converted to an input coin with $k = 0$:

$$\mathsf{Exc}(\mathsf{tx}) \coloneqq \sum\hat{\mathbf{C}} - \sum\mathbf{C} - \mathsf{Com.C}(\mathsf{cp},s;0)\ . \tag{5.4}$$

Intuitively, $\mathsf{Exc}(\mathsf{tx})$ should be a commitment to 0, as the committed input and output values of the transaction should cancel out; this is evidenced by giving a signature under key $\mathsf{Exc}(\mathsf{tx})$ (which could be represented as the sum of elements (E_i) due to aggregation; see below).

A transaction $\mathsf{tx} = (s, \mathbf{C}, \hat{\mathbf{C}}, K)$ with $K = (\pi, \mathbf{E}, \sigma)$ is said to be valid if all range proofs are valid, $\mathsf{Exc}(\mathsf{tx}) = \sum \mathbf{E}$, and σ is a valid signature for $\mathbf{E}$ (with all messages ε).[11]

When a user wants to make a payment of an amount ρ, she creates a transaction tx with input coins $\mathbf{C}$ of values $\mathbf{v}$ with $\sum \mathbf{v} \geq \rho$ and with output coins a list of fresh change coins of values $\mathbf{v}'$ so that $\sum \mathbf{v}' = \sum \mathbf{v} - \rho$. She also appends one more special coin of value ρ to the output. The pre-transaction ptx is then defined as this transaction tx, the remainder $\mathsf{ptx.rmdr} := \rho$ and the key for the special coin.

When receiving a pre-transaction $\mathsf{ptx} = (\mathsf{tx}, \rho, k)$, the receiver first checks that tx is valid and that k is a key for the special coin $C' := \mathsf{tx.out}[|\mathsf{tx.out}|]$ of value ρ. He then creates a transaction tx' that spends C' (using its key k) and creates coins of combined value ρ. Aggregating tx and tx' yields a transaction tx'' with $\mathsf{tx}''.\mathsf{sply} = 0$, $\mathsf{tx}''.\mathsf{in} = \mathsf{ptx.in}$ and $\mathsf{tx}''.\mathsf{out}$ containing $\mathsf{ptx.chg}$ and the freshly created coins. The receiver then submits tx'' to the ledger.

The ledger accepts a transaction if it is valid (as defined above) and if its input coins are contained in the output coin list of the ledger (which corresponds to the UTXO set in other systems). We do not consider any other conditions related to the consensus mechanism, such as fees being included in a transaction to incentivize its inclusion in the ledger or a proof-of-work being included in a minting transaction.

In Fig. 5.7 we first define auxiliary algorithms that create coins and transactions and verify their validity by instantiating the procedures V and Cons from Definition 5.5. Using these we then formally define $\mathsf{MW}[\mathsf{Com}, \mathsf{Sig}, \Pi]$ in Fig. 5.8.

Lemma 5.9 (Collision-resistance)**.** *Let* Com *be a (binding and hiding) commitment scheme. Then for any* $(v_0, v_1) \in \mathcal{V}_{\mathsf{cp}}^2$*, the probability that* $\mathsf{C}(\mathsf{cp}, v_0; r_0) = \mathsf{C}(\mathsf{cp}, v_1; r_1)$ *for* $r_0, r_1 \leftarrow\!\!\$\, \mathcal{R}_{\mathsf{cp}}$ *is negligible.*

The proof of the lemma is straightforward: for $v_0 \neq v_1$ this would break binding and for $v_0 = v_1$ it would break hiding.

Correctness. We start with showing some properties of the auxiliary algorithms in Fig. 5.7. For any $\mathbf{v} \in [\![0, v_{\max}]\!]^*$ and $(\mathbf{C}, \mathbf{k}, \pi) \leftarrow \mathsf{Coin}(\mathsf{pp}, \mathbf{v})$, we have $\mathsf{Cons}(\mathsf{pp}, \mathbf{C}, \mathbf{v}, \mathbf{k})$ with overwhelming probability due to Lemma 5.9. Moreover, correctness of Sig and Π implies that MkTx run on consistent $(\mathbf{C}, \mathbf{v}, \mathbf{k})$ and values $\hat{\mathbf{v}} \in [\![0, v_{\max}]\!]^*$ with $\sum \hat{\mathbf{v}} \geq \sum \mathbf{v}$ produces a tx which is accepted by V with overwhelming probability and whose supply is the difference $\sum \mathbf{v} - \sum \hat{\mathbf{v}}$.

We now show that the protocol $\mathsf{MW}[\mathsf{Com}, \mathsf{Sig}, \Pi]$ described in Fig. 5.8 satisfies Definition 5.5. It is immediate that an empty ledger output by $\mathsf{G}(1^\lambda, v_{\max})$ verifies. As M invokes MkTx on empty inputs and output values $\mathbf{v}$, correctness of M follows from correctness of MkTx. Correctness of S also follows from correctness of MkTx when the preconditions on the values, consistency of the coins and the supply, and $\sum \mathbf{v} - \sum \mathbf{v}' = \rho \in [\![0, v_{\max}]\!]$ hold (note that $\mathsf{ptx.rmdr} = \rho$). Therefore, with overwhelming probability the pre-transaction is valid, and the change coins are consistent. Correctness of A is straightforward: it returns a transaction with the desired supply, input, and output coin list whose validity follows from correctness of $\mathsf{Sig.A}$ and $\Pi.\mathsf{V}$ and $\sum \mathbf{E}_0 + \sum \mathbf{E}_1 = \hat{\mathbf{C}}_0 - \sum \mathbf{C}_0 - \mathsf{C}(\mathsf{cp}, s_0, 0) + \hat{\mathbf{C}}_1 - \sum \mathbf{C}_1 - \mathsf{C}(\mathsf{cp}, s_1, 0) = \sum \hat{\mathbf{C}} - \sum \mathbf{C} - \mathsf{C}(\mathsf{cp}, s_0 + s_1, 0)$, where the first equation follows from $\mathsf{V}(\mathsf{pp}, \mathsf{tx}_0)$ and $\mathsf{V}(\mathsf{pp}, \mathsf{tx}_0)$ and the second from the properties of cut-through.

For any adversary $\mathcal{A}_{\mathsf{L}}$ returning (Λ, tx), if $\mathsf{V}(\mathsf{pp}, \Lambda) = 1$, then $\Lambda.\mathsf{in} = (\,)$ and Λ is valid when interpreted as a transaction. Since the input list of Λ is empty, $\mathsf{L}(\mathsf{pp}, \Lambda, \mathsf{tx}) = \mathsf{A}(\mathsf{pp}, \Lambda, \mathsf{tx})$ and so L is correct because A is.

Finally, we consider R, which is slightly more involved. Consider an adversary $\mathcal{A}_{\mathsf{R}}$ returning $(\mathsf{ptx}, \mathbf{v}'')$ with $\mathsf{ptx} = (\mathsf{tx}, \rho, k')$ and let $(\mathsf{tx}'', \mathbf{k}'') \leftarrow \mathsf{MW.R}(\mathsf{pp}, \mathsf{ptx}, \mathbf{v}'')$. First, the preconditions trivially guarantee that the output is not $\bot$. Consider the call $(\mathsf{tx}', \mathbf{k}'') \leftarrow \mathsf{MW.MkTx}(\mathsf{pp}, (C', \rho, k'), \mathbf{v}'')$

[11] If $\mathbf{E}$ in a transaction tx consists of a single element, it must be $E = \mathsf{Exc}(\mathsf{tx})$, so E could be omitted from the transaction; we keep it for consistency.

$\underline{\mathsf{MW.G}(1^\lambda, v_{\max})}$

$\Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$
$\mathsf{cp} \leftarrow \mathsf{Com.G}(\Gamma)$
$\mathsf{sp} \leftarrow \mathsf{Sig.G}(\Gamma)$
$(\mathsf{crs}, \tau) \leftarrow \Pi.\mathsf{G}(\Gamma, v_{\max})$
$\Lambda \coloneqq \big(0, (\,), (\,), ((\,), (\,), \mathbf{e})\big)$
return $\big(\mathsf{pp} \coloneqq (\mathsf{cp}, \mathsf{sp}, \mathsf{crs}), \Lambda\big)$

$\underline{\mathsf{MW.A}(\mathsf{pp}, \mathsf{tx}_0, \mathsf{tx}_1)}$

if $\neg\, \mathsf{MW.V}(\mathsf{pp}, \mathsf{tx}_0)$ **or**
  $\neg\, \mathsf{MW.V}(\mathsf{pp}, \mathsf{tx}_1)$ **or**
  $\mathsf{tx}_0.\mathsf{in} \cap \mathsf{tx}_1.\mathsf{in} \neq (\,)$ **or**
  $\mathsf{tx}_0.\mathsf{out} \cap \mathsf{tx}_1.\mathsf{out} \neq (\,)$
    return $\perp$
$\big(s_0, \mathbf{C}_0, \hat{\mathbf{C}}_0, (\pi_0, \mathbf{E}_0, \sigma_0)\big) \coloneqq \mathsf{tx}_0$
$\big(s_1, \mathbf{C}_1, \hat{\mathbf{C}}_1, (\pi_1, \mathbf{E}_1, \sigma_1)\big) \coloneqq \mathsf{tx}_1$
$\mathbf{C} \coloneqq \mathbf{C}_0 \,\|\, \mathbf{C}_1 - \hat{\mathbf{C}}_0 \,\|\, \hat{\mathbf{C}}_1$
$\hat{\mathbf{C}} \coloneqq \hat{\mathbf{C}}_0 \,\|\, \hat{\mathbf{C}}_1 - \mathbf{C}_0 \,\|\, \mathbf{C}_1$
$\pi \coloneqq (\pi_{0,i})_{i \in I_0} \,\|\, (\pi_{1,i})_{i \in I_1}$
  where $I_j \coloneqq \{i : \hat{C}_{j,i} \notin \mathbf{C}_{1-j}\}$
// π contains the proofs for coins in $\hat{\mathbf{C}}$
$\sigma \leftarrow \mathsf{Sig.A}\big(\mathsf{sp}, (\mathbf{E}_0, \sigma_0), (\mathbf{E}_1, \sigma_1)\big)$
$K \coloneqq (\pi, \mathbf{E}_0 \,\|\, \mathbf{E}_1, \sigma)$
return $(s_0 + s_1, \mathbf{C}, \hat{\mathbf{C}}, K)$

$\underline{\mathsf{MW.M}(\mathsf{pp}, \hat{\mathbf{v}})}$

$(\mathsf{tx}, \hat{\mathbf{k}}) \leftarrow \mathsf{MW.MkTx}(\mathsf{pp}, ((\,), (\,), (\,)), \hat{\mathbf{v}})$
return $(\mathsf{tx}, \hat{\mathbf{k}})$ // If $\perp \leftarrow \mathsf{MkTx}$, M returns $\perp$

$\underline{\mathsf{MW.S}(\mathsf{pp}, (\mathbf{C}, \mathbf{v}, \mathbf{k}), \mathbf{v}')}$

$\rho \coloneqq \sum \mathbf{v} - \sum \mathbf{v}'$
$(\mathsf{tx}, \hat{\mathbf{k}}) \leftarrow \mathsf{MW.MkTx}(\mathsf{pp}, (\mathbf{C}, \mathbf{v}, \mathbf{k}), \mathbf{v}' \,\|\, \rho)$
$\mathsf{ptx} \coloneqq (\mathsf{tx}, \rho, \hat{k}_{|\mathbf{v}'|+1})$
return $(\mathsf{ptx}, (\hat{k}_i)_{i=1}^{|\mathbf{v}'|})$

$\underline{\mathsf{MW.R}(\mathsf{pp}, \mathsf{ptx}, \mathbf{v}'')}$

$(\mathsf{tx}, \rho, k') \coloneqq \mathsf{ptx}$
if $\neg\, \mathsf{MW.V}(\mathsf{pp}, \mathsf{ptx})$ **or** $\rho \neq \sum \mathbf{v}''$
  return $\perp$
$C' \coloneqq \mathsf{tx.out}[|\mathsf{tx.out}|]$
$(\mathsf{tx}', \mathbf{k}'') \leftarrow \mathsf{MW.MkTx}(\mathsf{pp}, (C', \rho, k'), \mathbf{v}'')$
$\mathsf{tx}'' \leftarrow \mathsf{MW.A}(\mathsf{pp}, \mathsf{tx}, \mathsf{tx}')$
return $(\mathsf{tx}'', \mathbf{k}'')$

$\underline{\mathsf{MW.L}(\mathsf{pp}, \Lambda, \mathsf{tx})}$

if $\Lambda.\mathsf{in} \neq (\,)$ **or** $\mathsf{tx.in} \not\subseteq \Lambda.\mathsf{out}$
  return $\perp$
return $\mathsf{MW.A}(\mathsf{pp}, \Lambda, \mathsf{tx})$ // returns $\perp$ if Λ or tx invalid

Figure 5.8: The MW aggregate cash system. (Recall that algorithms return $\perp$ when one of their subroutines returns $\perp$.)

inside MW.R. We claim that with overwhelming probability, $(\mathsf{tx.in} \,\|\, \mathsf{tx}'.\mathsf{in}) \cap (\mathsf{tx.out} \,\|\, \mathsf{tx}'.\mathsf{out}) = (C')$. First, $\mathsf{tx.in} \cap \mathsf{tx.out} = (\,)$, as otherwise $\mathsf{V}(\mathsf{pp}, \mathsf{tx}) = 0$ and $\mathsf{V}(\mathsf{pp}, \mathsf{ptx}) = 0$. By definition of MkTx, $\mathsf{tx}'.\mathsf{in} = (C')$ and by Lemma 5.9, $\mathsf{tx}'.\mathsf{out} \cap (\mathsf{tx.in} \,\|\, (C')) = (\,)$ with overwhelming probability. Hence:

$$(\mathsf{tx.in} \,\|\, \mathsf{tx}'.\mathsf{in}) \cap (\mathsf{tx.out} \,\|\, \mathsf{tx}'.\mathsf{out}) = (C') \cap \mathsf{tx.out} = (C')$$

and by correctness of A, C' is the only coin removed by cut-through during the call $\mathsf{tx}'' \leftarrow \mathsf{MW.A}(\mathsf{pp}, \mathsf{tx}, \mathsf{tx}')$. Thus, the input coin list of tx'' is the same as that of ptx and the change is contained in the output coin list of tx''. The pre-conditions $\mathsf{V}(\mathsf{pp}, \mathsf{ptx})$ and $\sum \mathbf{v}'' = \rho$ imply that $\mathsf{tx.sply} = 0$ and $\mathsf{tx}'.\mathsf{sply} = 0$, respectively. Hence, $\mathsf{tx}''.\mathsf{sply} = 0$ by correctness of A. Validity of tx'' and consistency of the new coins follow from correctness of A (and validity of the output of MkTx).

5.4 Inflation resistance

Theorem 5.10 (Inflation-resistance (Def. 5.6))**.** *Assume that the pair* (Com, Sig) *is EUF-NZO-secure and that* Π *is zero-knowledge and simulation-extractable. Then the aggregate cash system*

Game Game_0

$\Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$; $\mathsf{cp} \leftarrow \mathsf{Com.G}(\Gamma)$; $\mathsf{sp} \leftarrow \mathsf{Sig.G}(\Gamma)$
$(\mathsf{crs}, \tau) \leftarrow \Pi.\mathsf{G}(\Gamma, v_{\max})$; $\mathsf{pp} := (\mathsf{cp}, \mathsf{sp}, \mathsf{crs})$; $\Lambda_0 := \big(0, (\,), (\,), ((\,), (\,), \mathbf{e})\big)$
$(\Lambda, \mathsf{ptx}, \mathbf{v}) \leftarrow \mathcal{A}(\mathsf{pp}, \Lambda_0)$
$(\mathsf{tx}, \mathbf{k}) \leftarrow \mathsf{MW.R}(\mathsf{pp}, \mathsf{ptx}, \mathbf{v})$
return $\mathsf{V}(\mathsf{pp}, \Lambda)$ **and** $\mathsf{V}(\mathsf{pp}, \mathsf{tx})$ **and** $\Lambda.\mathsf{in} = (\,)$
and $\mathsf{tx.in} \subseteq \Lambda.\mathsf{out}$ **and** $\Lambda.\mathsf{sply} < \sum \mathbf{v}$

Games Game_1 and $\boxed{\mathrm{Game}_2}$

$\Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$; $\mathsf{cp} \leftarrow \mathsf{Com.G}(\Gamma)$; $\mathsf{sp} \leftarrow \mathsf{Sig.G}(\Gamma)$
$(\mathsf{crs}, \tau) \leftarrow \Pi.\mathsf{G}(\Gamma, v_{\max})$; $\mathsf{pp} := (\mathsf{cp}, \mathsf{sp}, \mathsf{crs})$; $\Lambda_0 := \big(0, (\,), (\,), ((\,), (\,), \mathbf{e})\big)$
$(\Lambda, \mathsf{ptx}, \mathbf{v}) \leftarrow \mathcal{A}(\mathsf{pp}, \Lambda_0)$
$(\mathsf{tx}, \mathbf{k}) \leftarrow \mathsf{MW.R}(\mathsf{pp}, \mathsf{ptx}, \mathbf{v})$
$\boxed{\textbf{if } p/v_{\max} \le |\mathbf{v}| + |\Lambda.\mathsf{out}| + |\mathsf{ptx.chg}| : \textbf{return } 0}$ (I)
return $\mathsf{V}(\mathsf{pp}, \Lambda)$ **and** $\mathsf{V}(\mathsf{pp}, \mathsf{tx})$ **and** $\Lambda.\mathsf{in} = (\,)$
and $\mathsf{ptx.in} \subseteq \Lambda.\mathsf{out}$ **and** $\Lambda.\mathsf{sply} < \mathsf{ptx.rmdr}$ **and** $\mathsf{V}(\mathsf{pp}, \mathsf{ptx})$

Game $\boxed{\mathrm{Game}_4}$

$\Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$; $\mathsf{cp} \leftarrow \mathsf{Com.G}(\Gamma)$; $\mathsf{sp} \leftarrow \mathsf{Sig.G}(\Gamma)$
$(\mathsf{crs}, \tau) \leftarrow \Pi.\mathsf{G}(\Gamma, v_{\max})$; $\mathsf{pp} := (\mathsf{cp}, \mathsf{sp}, \mathsf{crs})$; $\Lambda_0 := \big(0, (\,), (\,), ((\,), (\,), \mathbf{e})\big)$
$(\Lambda, \mathsf{ptx}, \mathbf{v}) \leftarrow \mathcal{A}(\mathsf{pp}, \Lambda_0)$
$(s, \mathbf{C}_\Lambda, \hat{\mathbf{C}}, K) := \Lambda$; $(\pi, \mathbf{E}, \sigma) := K$ // just parsing
$(\mathsf{tx}', \rho, k^*) := \mathsf{ptx}$; $(s', \mathbf{C}, \mathbf{C}' \| (C^*), K') := \mathsf{tx}'$; $(\pi' \| (\pi^*), \mathbf{E}', \sigma') := K'$ // just parsing
$(\mathsf{tx}, \mathbf{k}) \leftarrow \mathsf{MW.R}(\mathsf{pp}, \mathsf{ptx}, \mathbf{v})$
if $p/v_{\max} \le |\mathbf{v}| + |\hat{\mathbf{C}}| + |\mathbf{C}'|$: **return** 0
$\boxed{\begin{array}{l}(\hat{\mathbf{v}} \| \mathbf{v}', \hat{\mathbf{k}} \| \mathbf{k}') := \Pi.\mathsf{Ext}(\mathsf{crs}, \tau, (\mathsf{cp}, \hat{\mathbf{C}} \| \mathbf{C}'), \pi \| \pi') \\ \textbf{if } \neg \mathsf{R}_{v_{\max}}\big((\mathsf{cp}, \hat{\mathbf{C}} \| \mathbf{C}'), (\hat{\mathbf{v}} \| \mathbf{v}', \hat{\mathbf{k}} \| \mathbf{k}')\big) : \textbf{return } 0\end{array}}$
return $\mathsf{V}(\mathsf{pp}, \Lambda)$ **and** $\mathsf{V}(\mathsf{pp}, \mathsf{tx})$ **and** $\mathbf{C}_\Lambda = (\,)$
and $\mathbf{C} \subseteq \hat{\mathbf{C}}$ **and** $s < \rho$ **and** $\mathsf{V}(\mathsf{pp}, \mathsf{ptx})$

Figure 5.9: Games modifying $\mathrm{INFL}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max})$. Changes w.r.t. previous games are highlighted or $\boxed{\text{boxed}}$.

$\mathsf{MW}[\mathsf{Com}, \mathsf{Sig}, \Pi]$ *is secure against inflation. More precisely, for any* $v_{\max}$ *and any* PPT *adversary* $\mathcal{A}$*, there exists a negligible function* $\nu_{\mathcal{A}}$ *and* PPT *adversaries* $\mathcal{B}$*,* $\mathcal{B}_{zk}$ *and* $\mathcal{B}_{se}$ *such that*

$$\mathsf{Adv}^{\mathrm{infl}}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) \le \mathsf{Adv}^{\text{euf-nzo}}_{\mathsf{Com},\mathsf{Sig},\mathcal{B}}(\lambda) + \mathsf{Adv}^{\text{s-ext}}_{\Pi,\mathsf{R}_{v_{\max}},\mathcal{B}_{se}}(\lambda) + \nu_{\mathcal{A}}(\lambda) \ .$$

Proof. Consider an adversary $\mathcal{A}$ that runs on input public parameters $\mathsf{pp} = (\mathsf{cp}, \mathsf{sp}, \mathsf{crs})$ and an empty ledger, and wins the game $\mathrm{INFL}_{\mathcal{A},\mathsf{MW}}(\lambda, v_{\max})$ with non-negligible probability. The proof proceeds via the following sequence of games, defined in Fig. 5.9 (and whose indistinguishability we argue below):

Game$_0$. This is the original inflation game as presented in Definition 5.6 for the specific instantiation $\mathsf{Cash} \coloneqq \mathsf{MW}$ where $\mathsf{MW.G}$ has been written out and the first winning condition $\bot \notin \mathsf{MW.L}(\mathsf{pp}, \Lambda, \mathsf{tx})$ has been expanded with all those that are necessary for the specific case of MW. Hence,

$$\mathsf{Adv}^{\text{Game}_0}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) = \mathsf{Adv}^{\text{infl}}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) \ . \tag{5.5}$$

Game$_1$. We strenghten the previous game by adding an extra winning condition. We claim that this is perfectly indistinguishable from Game$_0$. Since $\mathsf{V}(\mathsf{pp}, \mathsf{tx}) = 1$, this implies in particular that $\mathsf{MW.R}$ did not return $\bot$. Then, $\mathsf{tx.in} = \mathsf{ptx.in}$ by correctness of $\mathsf{MW.R}$ and, by inspection of $\mathsf{MW.R}$, $\mathsf{ptx.rmdr} = \sum \mathbf{v}$ and $\mathsf{V}(\mathsf{pp}, \mathsf{ptx})$ hold whenever $\mathsf{MW.R}$ does not return $\bot$. Hence,

$$\mathsf{Adv}^{\text{Game}_1}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) = \mathsf{Adv}^{\text{Game}_0}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) \ . \tag{5.6}$$

Game$_2$. We modify Game$_1$ so that the experiment returns 0 whenever the adversary creates too many coins (which may cause the sum of their values to be larger than p). We claim that the two games are computationally indistinguishable. Since the adversary $\mathcal{A}$ runs in polynomial time, there is a polynomial $t_{\mathcal{A}}(\lambda)$ that upper-bounds the output length of $\mathcal{A}$, and in particular $|\mathbf{v}| + |\Lambda.\mathsf{out}| + |\mathsf{ptx.chg}|$. On the other hand, the value space $\mathcal{V}_{\mathsf{cp}} = \mathbb{Z}_p$ is such that $\lfloor \log p \rfloor + 1 = \lambda$; in other words, $p \geq 2^{\lambda-1}$. Therefore, there exists $\lambda_0 \in \mathbb{N}$ such that:

$$v_{\max} \cdot t_{\mathcal{A}}(\lambda) < 2^{\lambda-1} \leq p \qquad \text{for any } \lambda \geq \lambda_0.$$

Let $\nu_{\mathcal{A}}(\lambda)$ denote the probability that Game$_2$ returns 0 in line (I). We thus have

$$\mathsf{Adv}^{\text{Game}_2}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) \geq \mathsf{Adv}^{\text{Game}_1}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) - \nu_{\mathcal{A}}(\lambda) \ . \tag{5.7}$$

Since there exists $\lambda_0 \in \mathbb{N}$ such that $|\mathbf{v}| + |\Lambda.\mathsf{out}| + |\mathsf{ptx.chg}| \leq t_{\mathcal{A}}(\lambda) < p/v_{\max}$ for all $\lambda > \lambda_0$, we have $\nu_{\mathcal{A}}(\lambda) = 0$ for all such λ. Therefore, $\nu_{\mathcal{A}}(\lambda)$ is negligible.

Game$_4$. This final game attempts to extract valid openings for $\hat{\mathbf{C}}$ (the output coins of the ledger) and $\mathbf{C}'$ (the change coins of the pre-transaction) from the proofs contained in the kernels of Λ and tx'; it returns 0 if extraction fails for any of these coins. The rest of the game is left unchanged. Consider an adversary $\mathcal{B}_{se}$ for game S-EXT$_{\Pi}$ which on input a simulated CRS crs simulates Game$_3$ for $\mathcal{A}$ (again this does not require to query oracle PROVE) and returns $(\hat{\mathbf{C}} \| \mathbf{C}', \pi \| \pi')$ if Game$_3$ returns 1 and aborts otherwise. Note that for Game$_3$ to return 1, all range proofs in the kernel of Λ and tx' must be valid. Hence, $\mathcal{B}_{se}$ wins game S-EXT$_{\Pi}$ exactly when Game$_3$ returns 1 and Game$_4$ returns 0, so that

$$\mathsf{Adv}^{\text{Game}_4}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) = \mathsf{Adv}^{\text{Game}_3}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) - \mathsf{Adv}^{\text{s-ext}}_{\Pi,\mathsf{R}_{v_{\max}},\mathcal{B}_{se}}(\lambda) \ . \tag{5.8}$$

The reduction to EUF-NZO. We now construct an adversary $\mathcal{B}$ against EUF-NZO-security of $(\mathsf{Com}, \mathsf{Sig})$. $\mathcal{B}$ takes as input $(\mathsf{cp}, \mathsf{sp})$ and simulates Game$_4$: it retrieves Γ from $(\mathsf{cp}, \mathsf{sp})$, generates $\mathsf{crs} \leftarrow \Pi.\mathsf{Sim.G}(\Gamma)$, and runs $\mathcal{A}$ on input $(\mathsf{cp}, \mathsf{sp}, \mathsf{crs})$ and an empty ledger. If the game returns 0, then $\mathcal{B}$ aborts. Otherwise, we show that $\mathcal{B}$ can break EUF-NZO each time Game$_4$ returns 1.

Game$_4$ returning 1 implies in particular (in all the following we use the notation of Fig. 5.9):

(i) $\mathsf{V}(\mathsf{pp}, \Lambda) = 1$; (ii) $\mathbf{C}_\Lambda = (\,)$; (iii) $\mathbf{C} \subseteq \hat{\mathbf{C}}$; (iv) $s < \rho$;

(v) $(|\mathbf{v}| + |\hat{\mathbf{C}}| + |\mathbf{C}'|) \cdot v_{\max} < p$;

(vi) $\mathsf{V}(\mathsf{pp}, \mathsf{ptx}) = 1$, which in turn implies:

(a) $\mathsf{V}(\mathsf{pp}, \mathsf{tx}') = 1$;

(b) $s' \coloneqq \mathsf{tx}'.\mathsf{sply} = 0$;

(c) $C^* = \mathsf{Com.C}(\mathsf{cp}, \rho; k^*)$.

Let $\hat{v} \coloneqq \sum \hat{\mathbf{v}}$ and $\hat{k} \coloneqq \sum \hat{\mathbf{k}}$. Two cases may occur.

- $\hat{v} \not\equiv s \bmod p$: In this case, $\mathcal{B}$ returns $(\mathbf{E}, \sigma, (\hat{v} - s, \hat{k}))$. We claim that this is a valid EUF-NZO solution. By (i), σ is a valid signature for $\mathbf{E}$ and $\sum \mathbf{E} = \mathsf{Exc}(\Lambda)$ where by definition:

$$\begin{aligned}
\mathsf{Exc}(\Lambda) &= \textstyle\sum \hat{\mathbf{C}} - \sum \mathbf{C}_\Lambda - \mathsf{Com.C}(\mathsf{cp}, s; 0) \\
&= \textstyle\sum \hat{\mathbf{C}} - \mathsf{Com.C}(\mathsf{cp}, s; 0) && \text{(by (ii))} \\
&= \textstyle\sum_{i=1}^{|\hat{\mathbf{C}}|} \mathsf{Com.C}(\mathsf{cp}, \hat{v}_i; \hat{k}_i) - \mathsf{Com.C}(\mathsf{cp}, s; 0) \\
&= \mathsf{Com.C}(\mathsf{cp}, \hat{v} - s; \hat{k})\ .
\end{aligned}$$

  Since $\hat{v} - s \not\equiv 0 \bmod p$, $(\hat{v} - s, \hat{k})$ is a non-zero opening for $\sum \mathbf{E}$.

- $\hat{v} \equiv s \bmod p$: In this case, $\mathcal{B}$ must exploit the pre-transaction (unlike in the previous case where the ledger was sufficient). Let $v' \coloneqq \sum \mathbf{v}'$ and $k' \coloneqq \sum \mathbf{k}'$. Let us denote with I the set of indexes such that $\mathbf{C} = (\hat{\mathbf{C}}[i])_{i \in I}$, which exists by (iii). By (vi)(a), σ' is a valid signature for $\mathbf{E}'$ and $\sum \mathbf{E}' = \mathsf{Exc}(\mathsf{tx}')$ where by definition:

$$\begin{aligned}
\mathsf{Exc}(\mathsf{tx}') &= \textstyle\sum \mathbf{C}' + C^* - \sum \mathbf{C} - \mathsf{Com.C}(\mathsf{cp}, s'; 0) \\
&= \textstyle\sum \mathbf{C}' + C^* - \sum \mathbf{C} && \text{(by (vi)(b))} \\
&= \textstyle\sum \mathbf{C}' + \mathsf{Com.C}(\mathsf{cp}, \rho; k^*) - \sum \mathbf{C} && \text{(by (vi)(c))} \\
&= \mathsf{Com.C}(\mathsf{cp}, v'; k') + \mathsf{Com.C}(\mathsf{cp}, \rho; k^*) - \textstyle\sum_{i \in I} \mathsf{Com.C}(\mathsf{cp}, \hat{v}_i; \hat{k}_i) \\
&= \mathsf{Com.C}\big(\mathsf{cp}, v' + \rho - \textstyle\sum_{i \in I} \hat{v}_i; k' + k^* - \sum_{i \in I} \hat{k}_i\big)\ ,
\end{aligned}$$

  Let $v'' \coloneqq v' + \rho - \sum_{i \in I} \hat{v}_i$ and $k'' \coloneqq k' + k^* - \sum_{i \in I} \hat{k}_i$, so that (v'', k'') is an opening of $\mathsf{Exc}(\mathsf{tx}')$. At this stage, we only need to prove that $v'' \not\equiv 0 \bmod p$. To see it, note that:

$$\rho \overset{\text{(iv)}}{>} s \geq \hat{v} = \sum_{i \in [\![1, |\hat{\mathbf{C}}|]\!]} \hat{v}_i \geq \sum_{i \in I} \hat{v}_i,$$

  where the second inequality follows from $s \equiv \hat{v} \pmod p$, $s \geq 0$ (by (i)), and $0 \leq \hat{v} < |\hat{\mathbf{C}}| \cdot v_{\max} < p$ (by (v)). Since $v'' \geq \rho - \sum_{i \in I} \hat{v}_i$, this implies that $v'' > 0$. On the other hand:

$$v'' \leq v' + \rho \leq (|\mathbf{C}'| + |\mathbf{v}|) \cdot v_{\max} < p$$

  again by (v). Hence, $0 < v'' < p$, and this proves our claim. Wrapping up, $(\mathbf{E}', \sigma', (v'', k''))$ is a valid EUF-NZO solution.

In both cases, $\mathcal{B}$ wins the EUF-NZO game every time Game_4 returns 1. We thus have:

$$\mathsf{Adv}^{\text{euf-nzo}}_{\mathcal{B}}(\lambda) \geq \mathsf{Adv}^{\text{Game}_4}_{\mathcal{A}}(\lambda, v_{\max})\ . \tag{5.9}$$

The theorem follows from Eqs. (5.5) to (5.9). □

5.5 Theft-resistance

Theorem 5.11 (Theft-resistance (Def. 5.7))**.** *Assume that the pair* $(\mathsf{Com}, \mathsf{Sig})$ *is EUF-CRO-secure and that* Π *is zero-knowledge and simulation-extractable. Then the aggregate cash system* $\mathsf{MW}[\mathsf{Com}, \mathsf{Sig}, \Pi]$ *is secure against coin theft. More precisely, for any* $v_{\max}$ *and any* PPT *adversary* $\mathcal{A}$*, which, via its oracle calls, makes the challenger create at most* $h_{\mathcal{A}}$ *coins and whose queries* $(\mathbf{C}, \mathbf{v}')$ *to* SEND *satisfy* $|\mathbf{v}'| \leq n_{\mathcal{A}}$*, there exists a negligible function* ν*, a* PPT *adversary* $\mathcal{B}$ *making a single signing query, and* PPT *adversaries* $\mathcal{B}_{zk}$ *and* $\mathcal{B}_{se}$ *such that*

$$\mathsf{Adv}^{\mathrm{steal}}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) \leq h_{\mathcal{A}}(\lambda) \cdot n_{\mathcal{A}}(\lambda) \cdot \left(\mathsf{Adv}^{\text{euf-cro}}_{\mathsf{Com},\mathsf{Sig},\mathcal{B}}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\Pi,\mathsf{R}_{v_{\max}},\mathcal{B}_{zk}}(\lambda) + \mathsf{Adv}^{\text{s-ext}}_{\Pi,\mathsf{R}_{v_{\max}},\mathcal{B}_{se}}(\lambda)\right) + \nu(\lambda) \ .$$

Proof. To simplify the analysis, we first modify game STEAL in that it aborts if the experiment generates the same coin twice by chance. By Lemma 5.9, the probability $\nu(\lambda)$ of this happening is negligible. Now consider an adversary $\mathcal{A}$ that wins the (modified) game STEAL in Fig. 5.5, thus $\mathsf{Hon} \not\subseteq \Lambda.\mathsf{out}$ holds when the adversary terminates. We proceed via a sequence of games.

Game$_0$. Inspection of the STEAL game shows the following:

- a coin in Hon must have been added to Hon during an oracle call to MINT, RECEIVE or LEDGER; during this, it is also added to $\Lambda.\mathsf{out}$;
- in order for a coin to be *removed* from $\Lambda.\mathsf{out}$, it has to be in $\mathsf{tx.in}$ for some tx queried to LEDGER;
- if after such a call the coin is still in Hon and the adversary stops, then it has won.

Following this analysis, we further modify the game STEAL as Game$_0$ in Fig. 5.10 (ignore the boxes for now), so that it declares $\mathcal{A}$ won whenever the condition $\mathsf{Hon} \not\subseteq \Lambda.\mathsf{out}$ is first satisfied. We have highlighted the changes w.r.t. the original game in gray. By the above analysis we have

$$\mathsf{Adv}^{\mathrm{Game}_0}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) \geq \mathsf{Adv}^{\mathrm{steal}}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) - \nu(\lambda) \ . \tag{5.10}$$

(Note that $\mathcal{A}$ could win Game$_0$ but not STEAL by putting a stolen coin back into the ledger.)

Game$_1$. To simplify the proof, we *strengthen* the security notion by defining a game that is easier to win than Game$_0$ and then show that even this is infeasible. Consider an adversary that queries SEND, which creates a pre-transaction ptx with change coins $\mathsf{ptx.chg}$, and then queries LEDGER on a transaction tx that spends coins of $\mathsf{ptx.chg}$ that are not in Hon yet. In the original game, this does not constitute a win, since only coins in Hon can be stolen.

Our strengthened game Game$_1$ does consider such behavior as winning the game. In particular, the game stores the change coins generated during SEND calls in a list Chg and removes them from Chg once they are added to Hon. It also stops and declares the game won if the adversary manages to spend a coin from Chg. Game$_1$ is also defined in Fig. 5.10 by including the boxes. Since every adversary that wins Game$_0$ also wins Game$_1$, we have:

$$\mathsf{Adv}^{\mathrm{Game}_1}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) \geq \mathsf{Adv}^{\mathrm{Game}_0}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) \ . \tag{5.11}$$

Inspection of Game$_1$ yields that at any point during the execution the following holds: $\mathsf{Chg} \cup \mathsf{Hon}$ contains exactly all coins ever produced by the game and $\mathsf{Chg} \cap \mathsf{Hon} = (\,)$: coins are produced by MINT, RECEIVE or SEND, where the former two add the coins to Hon and the latter adds them to Chg; further, all coins removed from Chg by LEDGER are added to Hon.

Games Game_0 and $\boxed{\mathsf{Game}_1}$

$(\mathsf{pp}, \Lambda) \leftarrow \mathsf{Cash.G}(1^\lambda, v_{\max})$
$\boxed{\mathsf{Chg},}\ \mathsf{Hon}, \mathsf{Val}, \mathsf{Key}, \mathsf{Ptx} := (\,)$
$\mathcal{A}^{\textsc{Mint},\textsc{Send},\textsc{Receive},\textsc{Ledger}}(\mathsf{pp}, \Lambda)$
return 0

Aux function $\mathsf{Store}(\mathbf{C}, \mathbf{v}, \mathbf{k})$

$\mathsf{Val}(\mathbf{C}) := \mathbf{v}\,;\ \mathsf{Key}(\mathbf{C}) := \mathbf{k}$

Oracle $\textsc{Mint}(\mathbf{v})$

$(\mathsf{tx}, \mathbf{k}) \leftarrow \mathsf{Cash.M}(\mathsf{pp}, \mathbf{v})$
$\Lambda \leftarrow \mathsf{Cash.L}(\mathsf{pp}, \Lambda, \mathsf{tx})$
$\mathsf{Hon} := \mathsf{Hon} \,\|\, \mathsf{tx.out}$
$\mathsf{Store}(\mathsf{tx.out}, \mathbf{v}, \mathbf{k})$
return tx

Oracle $\textsc{Send}(\mathbf{C}, \mathbf{v}')$

if $\mathbf{C} \not\subseteq \mathsf{Hon}$ **or** $\bigcup_{\mathsf{ptx}\in\mathsf{Ptx}} \mathsf{ptx.in} \cap \mathbf{C} \neq (\,)$
  return $\perp$
$(\mathsf{ptx}, \mathbf{k}') \leftarrow \mathsf{Cash.S}\big(\mathsf{pp}, \mathbf{C}, \mathsf{Val}(\mathbf{C}), \mathsf{Key}(\mathbf{C}), \mathbf{v}'\big)$
$\mathsf{Store}(\mathsf{ptx.chg}, \mathbf{v}', \mathbf{k}')\,;\ \mathsf{Ptx} := \mathsf{Ptx} \,\|\, (\mathsf{ptx})$
$\boxed{\mathsf{Chg} := \mathsf{Chg} \,\|\, (\mathsf{ptx.chg})}$
return ptx

Oracle $\textsc{Receive}(\mathsf{ptx}, \mathbf{v})$

$(\mathsf{tx}, \mathbf{k}) \leftarrow \mathsf{Cash.R}(\mathsf{pp}, \mathsf{ptx}, \mathbf{v})$
$\Lambda' \leftarrow \textsc{Ledger}(\mathsf{tx})$
if $\Lambda' = \perp$: **return** $\perp$
$\mathsf{Hon} := \mathsf{Hon} \,\|\, (\mathsf{tx.out} - \mathsf{ptx.chg})$
$\mathsf{Store}(\mathsf{tx.out} - \mathsf{ptx.chg}, \mathbf{v}, \mathbf{k})$
return tx

Oracle $\textsc{Ledger}(\mathsf{tx})$

$\Lambda' \leftarrow \mathsf{Cash.L}(\mathsf{pp}, \Lambda, \mathsf{tx})$
if $\Lambda' = \perp$: **return** $\perp$ **else** : $\Lambda := \Lambda'$
for all $\mathsf{ptx} \in \mathsf{Ptx}$ **do**
  if $\mathsf{ptx.chg} \subseteq \mathsf{tx.out}$:
    $\mathsf{Ptx} := \mathsf{Ptx} - (\mathsf{ptx})$
    $\mathsf{Hon} := (\mathsf{Hon} - \mathsf{ptx.in}) \,\|\, \mathsf{ptx.chg}$
    $\boxed{\mathsf{Chg} := \mathsf{Chg} - \mathsf{ptx.chg}}$
  $\boxed{\textbf{if } \mathsf{tx.in} \cap \mathsf{Chg} \neq (\,) \ \ /\!/ \text{ change stolen!}}$
  $\boxed{\textbf{stop game returning true}}$ (I)
if $\mathsf{tx.in} \cap \mathsf{Hon} \neq (\,)$ // coin stolen!
  stop game returning true (II)
return Λ

Figure 5.10: Reformulated and $\boxed{\text{strengthened}}$ coin stealing game.

Game_2. Consider a winning execution of Game_1 and let tx^* denote the transaction such that $\mathcal{A}$ wins during call $\textsc{Ledger}(\mathsf{tx}^*)$. Define a coin $\widetilde{C}$ as follows: if $\mathcal{A}$ stole a coin from Chg, i.e., it won in line (I), then $\widetilde{C}$ is the first coin in $\mathsf{tx}^*.\mathsf{in}$ that is also in Chg; if the adversary won by stealing a coin from Hon, i.e., in line (II), then $\widetilde{C}$ is the first coin in $\mathsf{tx}^*.\mathsf{in}$ that is also in Hon.

Now consider the case the adversary wins in line (II) *and* previously made a query $\textsc{Send}(\mathbf{C}, \mathbf{v}')$ with $\widetilde{C} \in \mathbf{C}$. Let $\widetilde{\mathsf{ptx}}$, with $\widetilde{C} \in \widetilde{\mathsf{ptx}}.\mathsf{in}$, be the pre-transaction returned by $\textsc{Send}$. Then we must have $(*)$ $\widetilde{\mathsf{ptx}}.\mathsf{chg} \not\subseteq \mathsf{tx}^*.\mathsf{out}$, as otherwise, during the final call to $\textsc{Ledger}$, Hon would have been updated to $\mathsf{Hon} := (\mathsf{Hon} - \widetilde{\mathsf{ptx}}.\mathsf{in}) \,\|\, \widetilde{\mathsf{ptx}}.\mathsf{chg}$, thereby removing $\widetilde{C}$ from Hon, which contradicts the definition of $\widetilde{C}$, as no coin can ever be re-added to Hon. We define $\tilde{\imath}$ as the index of the first coin in $\widetilde{\mathsf{ptx}}.\mathsf{chg}$ that is not in $\mathsf{tx}^*.\mathsf{out}$, which by $(*)$ exists.

Having now defined a coin $\widetilde{C}$, which uniquely exists for every winning execution of Game_1 and $\tilde{\imath}$, which uniquely exists if the adversary won in line (II) and queried $\mathbf{C} \ni \widetilde{C}$ to $\textsc{Send}$, we define Game_2. Let $h_{\mathcal{A}}$ and $n_{\mathcal{A}}$ be upper bounds on the number of coins created during the execution and on the number of change coins in a pre-transaction. Game_2 is defined like Game_1, except that at the beginning, Game_2 samples $\tilde{\imath} \leftarrow\!\!\$\, [n_{\mathcal{A}}]$ and guesses $\widetilde{C}$ among all (at most $h_{\mathcal{A}}$) produced coins and returns 0 in case the adversary lost *or* the guess was not correct. Game_2 for the cash system

MW is depicted in Fig. 5.11. Since the guess is uniform and perfectly hidden from the adversary, we have:

$$\mathsf{Adv}^{\mathrm{Game}_2}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) \geq \frac{1}{h_{\mathcal{A}} \cdot n_{\mathcal{A}}} \cdot \mathsf{Adv}^{\mathrm{Game}_1}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) \ . \tag{5.12}$$

Game$_3$. In Game$_3$ we introduce two modifications in how queries to SEND are handled (Fig. 5.11, including the boxes). First, the game returns 0 if the adversary makes a call $\textsc{Send}(\mathbf{C}, \mathbf{v}')$ with $\widetilde{C} \in \mathbf{C}$ and $|\mathbf{v}'| < \tilde{\imath}$. If there is a call $\mathsf{ptx} \leftarrow \textsc{Send}(\mathbf{C}, \mathbf{v}')$ with $\widetilde{C} \in \mathbf{C}$ and $|\mathbf{v}'| \geq \tilde{\imath}$, then the game defines $\overline{C} := \mathsf{ptx.chg}[\tilde{\imath}]$. Once $\overline{C}$ has been defined, the game returns 0 if the adversary makes a call $\textsc{Send}(\mathbf{C}, \mathbf{v}')$ with $\overline{C} \in \mathbf{C}$.

We claim that Game$_3$ returns 1 with exactly the same probability as Game$_2$, so that

$$\mathsf{Adv}^{\mathrm{Game}_3}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) = \mathsf{Adv}^{\mathrm{Game}_2}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) \ . \tag{5.13}$$

This is because the modifications only make the game return 0 *earlier*. To see this, we consider an execution of Game$_2$ which returns 1 in line (I) or (II) and show that the corresponding execution of Game$_3$ also returns 1. Consider first an execution which returns 1 in line (I). Since $\widetilde{C} \in \mathsf{Chg}$ and $\mathsf{Chg} \cap \mathsf{Hon} = (\,)$ (see argument after (5.11)), $\widetilde{C}$ is never queried to SEND, so Game$_3$ cannot return 0 in line (IV); moreover, $\overline{C}$ is never defined, so Game$_3$ cannot return 0 in line (III) either. Hence, Game$_3$ also returns 1 in line (I). Consider now an execution which returns 1 in line (II) during a query $\textsc{Ledger}(\mathsf{tx}^*)$. We consider several cases depending on the value of $\widetilde{\mathsf{ptx}}$ at the end of the execution:

1. $\widetilde{\mathsf{ptx}} = \bot$: the analysis is like in case (I): $\widetilde{C}$ is never queried to SEND (as otherwise $\widetilde{\mathsf{ptx}}$ would get defined) and $\overline{C}$ is never defined, hence Game$_3$ cannot return 0 in line (III) or (IV).

2. $\widetilde{\mathsf{ptx}} \neq \bot$, that is, $\widetilde{C}$ has been queried to SEND: First, since Game$_2$ arriving in line (II) implies that the $\tilde{\imath}$-th change output of $\widetilde{\mathsf{ptx}}$ exists, Game$_3$ cannot have returned 0 in line (IV). Hence, $\overline{C}$ was created and added to Chg. We first argue that $\overline{C}$ must still be in Chg at the end of the game: $\overline{C}$ can only be removed from Chg during an oracle call $\textsc{Ledger}(\mathsf{tx})$ with $\widetilde{\mathsf{ptx}}.\mathsf{chg} \subseteq \mathsf{tx.out}$. However, at the same time such a call removes $\widetilde{\mathsf{ptx}}.\mathsf{in}$ from Hon. This however contradicts that Game$_2$ returns 1 in line (II), which implies $\widetilde{C} \in \mathsf{Hon}$ when by construction we have $\widetilde{C} \in \widetilde{\mathsf{ptx}}.\mathsf{in}$. We conclude that $\overline{C} \in \mathsf{Chg}$ when the game returns. Hence $\overline{C}$ cannot have been queried to SEND (for which it must be in Hon) and therefore Game$_3$ does not return 0 in line (III).

Game$_4$. We define Game$_4$, a slight variation of Game$_3$. When $\widetilde{C}$ is created, instead of running $\Pi.\mathsf{P}$ on the witness (v, k), it sets $\pi_i \leftarrow \Pi.\mathsf{Sim}(\mathsf{crs}, \tau, (\mathsf{cp}, C_i))$; if $\mathcal{A}$ queries $\textsc{Send}(\mathbf{C}, \mathbf{v}')$ with $\widetilde{C} \in \mathbf{C}$ and there is an $\tilde{\imath}$-th change coin $\overline{C}$, it also simulates the proof for $\overline{C}$.

Game$_4$ is easily shown to be indistinguishable from Game$_3$ by constructing the following adversary $\mathcal{B}_{\mathrm{zk}}$ for game $\mathrm{ZK}_{\Pi,\mathsf{R}_{v_{\max}}}$: it receives crs (which is either produced by $\Pi.\mathsf{G}$ or by $\Pi.\mathsf{Sim}$) and simulates Game$_3$, querying its oracle $\textsc{Prove}((\mathsf{cp}, C), (v, k))$ when producing the proof for $C = \widetilde{C}$ or $C = \overline{C}$; $\mathcal{B}_{\mathrm{zk}}$ returns 1 if $\mathcal{A}$ wins Game$_3$ and 0 otherwise. Since $\mathcal{B}_{\mathrm{zk}}$ perfectly simulates Game$_3$ or Game$_4$ depending on the bit of its ZK challenger, we have:

$$\mathsf{Adv}^{\mathrm{Game}_4}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) \geq \mathsf{Adv}^{\mathrm{Game}_3}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) - \mathsf{Adv}^{\mathrm{zk}}_{\Pi,\mathsf{R}_{v_{\max}},\mathcal{B}_{\mathrm{zk}}}(\lambda) \ . \tag{5.14}$$

Games Game_2 and $\boxed{\mathsf{Game}_3}$

$\tilde{c} \leftarrow\$ [h_{\mathcal{A}}]$; $\tilde{\imath} \leftarrow\$ [n_{\mathcal{A}}]$; $c := 0$;
$\widetilde{C} := \bot$; $\widetilde{\mathsf{ptx}} := \bot$; $\boxed{\overline{C} := \bot}$
$(\mathsf{pp}, \Lambda) \leftarrow \mathsf{MW.G}(1^\lambda, v_{\max})$
$\mathsf{Chg}, \mathsf{Hon}, \mathsf{Val}, \mathsf{Key}, \mathsf{Ptx} := (\,)$
$\mathcal{A}^{\textsc{Mint},\textsc{Send},\textsc{Receive},\textsc{Ledger}}(\mathsf{pp}, \Lambda)$
return 0

Aux function $\mathsf{Store}(\mathbf{C}, \mathbf{v}, \mathbf{k})$

$\mathsf{Val}(\mathbf{C}) := \mathbf{v}$; $\mathsf{Key}(\mathbf{C}) := \mathbf{k}$

$\mathsf{MW.Coin}(\mathsf{pp}, \mathbf{v})$

for $i = 1 \ldots |\mathbf{v}|$ **do**
  $c := c + 1$ // increase coin ctr
  $C_i := \mathsf{Com.C}(\mathsf{cp}, v_i; k_i)$
  $\pi_i \leftarrow \Pi.\mathsf{P}(\mathsf{crs}, (\mathsf{cp}, C_i), (v_i, k_i))$
  if $c = \tilde{c}$: $\widetilde{C} := C_i$
return $\left((C_i)_{i=1}^{|\mathbf{v}|}, (k_i)_{i=1}^{|\mathbf{v}|}, (\pi_i)_{i=1}^{|\mathbf{v}|}\right)$

Oracle $\textsc{Send}(\mathbf{C}, \mathbf{v}')$

if $\mathbf{C} \not\subseteq \mathsf{Hon}$ **or** $\bigcup_{\mathsf{ptx} \in \mathsf{Ptx}} \mathsf{ptx.in} \cap \mathbf{C} \neq (\,)$
  return $\bot$
$(\mathsf{ptx}, \mathbf{k}') \leftarrow \mathsf{MW.S}\big(\mathsf{pp}, \mathbf{C}, \mathsf{Val}(\mathbf{C}), \mathsf{Key}(\mathbf{C}), \mathbf{v}'\big)$
if $\widetilde{C} \in \mathbf{C}$: $\widetilde{\mathsf{ptx}} := \mathsf{ptx}$ // this can only happen once
$\mathsf{Store}(\mathsf{ptx.chg}, \mathbf{v}', \mathbf{k}')$; $\mathsf{Ptx} := \mathsf{Ptx} \,\|\, (\mathsf{ptx})$
$\mathsf{Chg} := \mathsf{Chg} \,\|\, (\mathsf{ptx.chg})$; **return** ptx

$\mathsf{MW.S}(\mathsf{pp}, (\mathbf{C}, \mathbf{v}, \mathbf{k}), \mathbf{v}')$

$\rho := \sum \mathbf{v} - \sum \mathbf{v}'$
if $\neg\, \mathsf{Cons}(\mathsf{pp}, \mathbf{C}, \mathbf{v}, \mathbf{k})$: **return** $\bot$
if $\mathbf{v} \,\|\, \mathbf{v}' \,\|\, \rho \not\subseteq [0, v_{\max}]^*$: **return** $\bot$

if $\overline{C} \in \mathbf{C}$: **stop game returning false** (III)
if $\widetilde{C} \in \mathbf{C}$ // ptx being created will be $\widetilde{\mathsf{ptx}}$
  if $|\mathbf{v}'| < \tilde{\imath}$: **stop game returning false** (IV)
  $(\hat{\mathbf{C}}, \hat{\mathbf{k}}, \hat{\pi}) \leftarrow \mathsf{MW.Coin}(\mathsf{pp}, \mathbf{v}' \,\|\, \rho)$
  $\overline{C} := \hat{\mathbf{C}}[\tilde{\imath}]$

$\boxed{\textbf{else}:}$ $(\hat{\mathbf{C}}, \hat{\mathbf{k}}, \hat{\pi}) \leftarrow \mathsf{MW.Coin}(\mathsf{pp}, \mathbf{v}' \,\|\, \rho)$
$\sigma \leftarrow \mathsf{Sig.S}(\mathsf{sp}, \sum \hat{\mathbf{k}} - \sum \mathbf{k}, \mathbf{e})$
$\mathsf{tx} := \big(0, \mathbf{C}, \hat{\mathbf{C}}, (\hat{\pi}, \sum \hat{\mathbf{C}} - \sum \mathbf{C}, \sigma)\big)$
return $\big(\mathsf{ptx} := (\mathsf{tx}, \rho, \hat{k}_{|\mathbf{v}'|+1}), (\hat{k}_i)_{i=1}^{|\mathbf{v}'|}\big)$

Oracle $\textsc{Ledger}(\mathsf{tx})$

$\Lambda' \leftarrow \mathsf{MW.L}(\mathsf{pp}, \Lambda, \mathsf{tx})$
if $\Lambda' = \bot$: **return** $\bot$ **else** : $\Lambda := \Lambda'$
for all $\mathsf{ptx} \in \mathsf{Ptx}$ **do**
  if $\mathsf{ptx.chg} \subseteq \mathsf{tx.out}$
    $\mathsf{Ptx} := \mathsf{Ptx} - (\mathsf{ptx})$
    $\mathsf{Hon} := (\mathsf{Hon} - \mathsf{ptx.in}) \,\|\, \mathsf{ptx.chg}$
    $\mathsf{Chg} := \mathsf{Chg} - \mathsf{ptx.chg}$
if $\mathsf{tx.in} \cap \mathsf{Chg} \neq (\,)$
  if $\widetilde{C} = \mathsf{tx.in}[j]$ with $j = \min\{i \mid \mathsf{tx.in}[i] \in \mathsf{Chg}\}$
    stop game returning true (I)
  else: stop game returning false
if $\mathsf{tx.in} \cap \mathsf{Hon} \neq (\,)$
  if $\widetilde{C} = \mathsf{tx.in}[j]$ with $j = \min\{i \mid \mathsf{tx.in}[i] \in \mathsf{Hon}\}$
  $\wedge\, (\widetilde{\mathsf{ptx}} = \bot \vee \tilde{\imath} = \min\{i \mid \widetilde{\mathsf{ptx}}.\mathsf{chg}[i] \notin \mathsf{tx.out}\})$
    stop game returning true (II)
  else: stop game returning false
return Λ

Figure 5.11: Game_2 and Game_3 for $\mathsf{Cash} := \mathsf{MW}$, where oracles $\textsc{Mint}$ and $\textsc{Receive}$ are defined as in Fig. 5.10 and MW.G, MW.M, MW.L, and MW.R are defined as in Fig. 5.8. Changes from Game_1 to Game_2 are highlighted and changes from Game_2 to Game_3 are $\boxed{\text{boxed}}$.

Game_5. Our final game will be Game_5, which is defined as Game_4, except that if a call $\textsc{Ledger}(\mathsf{tx}^*)$ ends up in line (I) or (II), then instead of immediately returning 1, Game_5 does the following: let $\mathsf{tx}^* = (s, \mathbf{C}, \hat{\mathbf{C}}, (\hat{\pi}, \mathbf{E}, \sigma))$; let $\mathbf{C}' := \mathbf{C} - (\widetilde{C})$ and for all $\mathbf{C}'$ (for which we have $\mathbf{C}' \subseteq \Lambda.\mathsf{out}$; otherwise MW.L, and thus $\textsc{Ledger}$, would return $\bot$), it collects the corresponding proofs π' in the kernel of

the ledger and it runs:

- $(\mathbf{v}, \mathbf{k}) \coloneqq \Pi.\mathsf{Ext}(\mathsf{crs}, \tau, (\mathsf{cp}, \mathbf{C}'), \pi')$
- $(\hat{\mathbf{v}}, \hat{\mathbf{k}}) \coloneqq \Pi.\mathsf{Ext}(\mathsf{crs}, \tau, (\mathsf{cp}, \hat{\mathbf{C}}), \hat{\pi})$

If we have $\neg\, \mathsf{R}_{v_{\max}}((\mathsf{cp}, \mathbf{C}' \,\|\, \hat{\mathbf{C}}), (\mathbf{v} \,\|\, \hat{\mathbf{v}}, \mathbf{k} \,\|\, \hat{\mathbf{k}}))$ then Game_5 returns 0; otherwise, it returns 1.

We show that Game_5 is indistinguishable from Game_4. To start with, note that since $\mathsf{MW.L}(\mathsf{pp}, \Lambda, \mathsf{tx}^*)$ did not return $\perp$, both Λ and tx^* are valid and thus the following hold:

- $\Pi.\mathsf{V}(\mathsf{crs}, \mathbf{C}', \pi')$ and
- $\Pi.\mathsf{V}(\mathsf{crs}, \hat{\mathbf{C}}, \hat{\pi})$.

We construct an adversary $\mathcal{B}_{\text{se}}$ against simulation-extractability of Π, which receives a simulated CRS crs and has access to an oracle PROVE, as follows. Adversary $\mathcal{B}_{\text{se}}$ simulates Game_4, using its oracle PROVE (since it does not have the simulation trapdoor) for the simulated proofs for $\widetilde{C}$ and $\overline{C}$; if Game_4 ends up in lines (I) or (II), $\mathcal{B}_{\text{se}}$ returns $(\mathbf{C}' \,\|\, \hat{\mathbf{C}}, \pi' \,\|\, \hat{\pi})$; otherwise $\mathcal{B}_{\text{se}}$ aborts. Since $\mathcal{B}_{\text{se}}$ wins game S-EXT_Π whenever Game_4 returns 1 while Game_5 would not, we have

$$\mathsf{Adv}^{\text{Game}_5}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) \geq \mathsf{Adv}^{\text{Game}_4}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) - \mathsf{Adv}^{\text{s-ext}}_{\Pi,\mathsf{R}_{v_{\max}},\mathcal{B}_{\text{se}}}(\lambda)\ . \tag{5.15}$$

The reduction to EUF-CRO. We now construct an adversary $\mathcal{B}$ against EUF-CRO of $(\mathsf{Com}, \mathsf{Sig})$, which makes a single call to its SIGN' oracle, and show that $\mathcal{B}$ breaks EUF-CRO with the same probability as $\mathcal{A}$ wins Game_5. The modified procedures $\mathsf{MW.Coin}$, $\mathsf{MW.MkTx}$ (which uses $\mathsf{MW.Coin}$ and is used by MINT and RECEIVE) and $\mathsf{MW.S}$ are formally defined in Fig. 5.12 ($\mathsf{MW.Coin}$ now has an additional parameter j, which is set to $\perp$ by default).

$\mathcal{B}$ receives input $(\mathsf{cp}, \mathsf{sp}, C^*)$. It computes $(\mathsf{crs}, \tau) \leftarrow \Pi.\mathsf{G}(\Gamma, v_{\max})$ and picks random $\tilde{c}$ and $\tilde{\imath}$. It runs $\mathcal{A}$ on $\mathsf{pp} \coloneqq (\mathsf{cp}, \mathsf{sp}, \mathsf{crs})$ and simulates all of $\mathcal{A}$'s oracles as defined by Game_5 (cf. also Fig. 5.11), except that it *embeds* the challenge C^* into $\widetilde{C}$ and, in case $\widetilde{C}$ is queried to SEND, it also embeds C^* into the $\tilde{\imath}$-th change coin $\overline{C}$. If there is no $\tilde{\imath}$-th change coin, in particular, if there are *no* change coins, then $\mathcal{B}$ aborts (as Game_5 would return 0 anyway).

By "embedding" C^* into $\widetilde{C}$, we mean that instead of computing $\widetilde{C}$ as $\mathsf{C}(\mathsf{cp}, v, k)$, $\mathcal{B}$ sets $\widetilde{C} \coloneqq C^* + \mathsf{C}(\mathsf{cp}, v, k)$ and does thus not know $\widetilde{C}$'s actual key $\tilde{k} = k + r^*$, where $C^* = r^*G$.

$\mathcal{B}$ uses the zero-knowledge simulator to produce the range proofs for the two coins $\widetilde{C}$ and $\overline{C}$. Moreover, when $\widetilde{C}$ is created by MINT, SEND or RECEIVE, the transaction containing it as its j-th output must be signed. $\mathcal{B}$ uses its related-key signing oracle SIGN' to do this: letting $(k_i)_i$ and $(\hat{k}_i)_{i \neq j}$ be the keys of the inputs and other outputs of the transaction, $\mathcal{B}$ requires a signature under $\sum_{i \neq j} \hat{k}_i + (k_j + r^*) - \sum_i k_i$; it thus makes a query $\text{SIGN}'(\sum \hat{\mathbf{k}} - \sum \mathbf{k}, \mathbf{e})$ (this is the only time $\mathcal{B}$ makes a query).

Finally, consider the query $\text{SEND}(\mathbf{C}, \mathbf{v}')$ with $\mathbf{C}[j] = \widetilde{C}$ for some j, which would also require the signing key for $\widetilde{C}$ in order to compute the resulting pre-transaction:

$$\widetilde{\mathsf{ptx}} = \big(\mathsf{tx} = (0, \mathbf{C}, \hat{\mathbf{C}}, (\hat{\pi}, \textstyle\sum \hat{\mathbf{C}} - \sum \mathbf{C}, \sigma)), \rho, \hat{k}_{|\mathbf{v}|+1}\big)\ ,$$

where σ is a signature for the verification key $\sum \hat{\mathbf{C}} - \sum \mathbf{C}$. Since $\mathcal{B}$ embedded C^* in $\mathbf{C}$ via $\widetilde{C}$ and in $\hat{\mathbf{C}}$ via $\overline{C}$, the two occurrences cancel out and $\mathcal{B}$ knows the corresponding signing key. More precisely, $\sum \hat{\mathbf{k}} - \sum \mathbf{k}$ is the signing key for $\sum \hat{\mathbf{C}} - \sum \mathbf{C}$, since:

$$\begin{aligned}\textstyle\sum \hat{\mathbf{C}} - \sum \mathbf{C} &= \textstyle\sum_{i \neq j} \mathsf{Com.C}(\mathsf{cp}, \hat{v}_i; \hat{k}_i) + \big(C^* + \mathsf{Com.C}(\mathsf{cp}, v_j; k_j)\big) \\ &\qquad - \textstyle\sum_{i \neq \tilde{\imath}} \mathsf{Com.C}(\mathsf{cp}, v_i; k_i) - \big(C^* + \mathsf{Com.C}(\mathsf{cp}, v_{\tilde{\imath}}; k_{\tilde{\imath}})\big) \\ &= \mathsf{Com.C}(\mathsf{cp}, 0; \textstyle\sum_i \hat{k}_i - \sum k_i)\ .\end{aligned}$$

Note that there can only be one SEND query containing $\widetilde{C}$. The *only* other query which $\mathcal{B}$ cannot answer (as it lacks a necessary coin key) is $\text{SEND}(\mathbf{C}, \mathbf{v}')$ with $\overline{C} \in \mathbf{C}$. If this happens then $\mathcal{B}$ aborts. Hence, $\mathcal{B}$ perfectly simulates Game_5.

We now show how $\mathcal{B}$ computes a solution for the EUF-CRO challenge whenever $\mathcal{A}$ wins Game_5. Fig. 5.12 specifies $\mathcal{B}$'s simulation of the oracle LEDGER and its behavior in case $\mathcal{A}$ wins Game_5 (via the procedure Finalize). We claim that whenever $\mathsf{Finalize}(\mathsf{tx}^*)$ is called, the following holds:

(i) $\widetilde{C} \in \mathsf{tx}^*.\mathsf{in}$; (ii) $\widetilde{C} \notin \mathsf{tx}^*.\mathsf{out}$; (iii) $\overline{C} \notin \mathsf{tx}^*.\mathsf{in}$; (iv) $\overline{C} \notin \mathsf{tx}^*.\mathsf{out}$.

Property (i) is clearly necessary for $\mathsf{Finalize}(\mathsf{tx}^*)$ to be called. Property (ii) must hold as otherwise $\mathsf{tx}^*.\mathsf{in} \cap \mathsf{tx}^*.\mathsf{out} \neq (\,)$, which implies $\mathsf{V}(\mathsf{pp}, \mathsf{tx}^*) = 0$ and hence $\mathsf{MW.A}(\mathsf{pp}, \Lambda, \mathsf{tx}^*)$ (and hence $\mathsf{MW.L}(\mathsf{pp}, \Lambda, \mathsf{tx}^*)$) would return $\perp$. To prove (iii) and (iv), we distinguish two cases. Assume first that $\mathsf{Finalize}(\mathsf{tx}^*)$ is called in line (I). Since $\widetilde{C} \in \mathsf{Chg}$ and $\mathsf{Chg} \cap \mathsf{Hon} = (\,)$ (see argument after (5.11)), $\widetilde{C}$ has never been queried to SEND, thus $\overline{C}$ has never been defined and (iii) and (iv) trivially hold. Assume now that $\mathsf{Finalize}(\mathsf{tx}^*)$ is called in line (II). If $\widetilde{\mathsf{ptx}} = \perp$ then as before $\overline{C} = \perp$ and (iii) and (iv) trivially hold. If $\widetilde{\mathsf{ptx}} \neq \perp$, then necessarily (by inspection of the code) $\widetilde{\mathsf{ptx}}.\mathsf{chg}[\tilde{\imath}] = \overline{C} \notin \mathsf{tx}^*.\mathsf{out}$ and (iv) holds. It remains to prove (iii). As in the reasoning for Game_3, we have $\overline{C} \in \mathsf{Chg}$. Moreover, we have $\mathsf{tx}^*.\mathsf{in} \cap \mathsf{Chg} = (\,)$, since otherwise $\mathcal{B}$ would have returned in line (I) of Fig. 5.12. Together this implies $\overline{C} \notin \mathsf{tx}^*.\mathsf{in}$.

It is easily seen that all coins in $\Lambda.\mathsf{out}$ have a valid proof in the ledger's kernel (otherwise MW.L (and thus LEDGER) would not have included them in $\Lambda.\mathsf{out}$). The reduction can thus use the extractor to obtain the values and keys of the coins in $\mathbf{C} - (\widetilde{C})$: $(v_i)_{i \in [|\mathbf{C}|] \setminus (j)}$ and $(k_i)_{i \in [|\mathbf{C}|] \setminus (j)}$. From the proofs π contained in tx^*, it can moreover extract the values and keys of the output coins $\hat{\mathbf{C}}$: $(\hat{v}_i)_{i \in [|\hat{\mathbf{C}}|]}$ and $(\hat{k}_i)_{i \in [|\hat{\mathbf{C}}|]}$. Since $\mathsf{MW.V}(\mathsf{pp}, \mathsf{tx}^*)$, we have:

$$
\begin{aligned}
&\text{(a)} \qquad \mathsf{Sig.V}(\mathsf{sp}, \mathbf{E}, \sigma) \\
&\text{(b)} \qquad \textstyle\sum \mathbf{E} = \sum \hat{\mathbf{C}} - \sum \mathbf{C} - \mathsf{Com.C}(\mathsf{cp}, s; 0) \\
&\qquad\qquad = \textstyle\sum_i \hat{C}_i - (\sum_{i \neq j} C_i + C^* + \mathsf{Com.C}(\mathsf{cp}, v_j; k_j)) - \mathsf{Com.C}(\mathsf{cp}, s; 0) \\
&\qquad\qquad = -C^* + \mathsf{Com.C}(\mathsf{cp}, \underbrace{\textstyle\sum_i \hat{v}_i - \sum_i v_i - s}_{=:v}; \underbrace{\textstyle\sum_i \hat{k}_i - \sum_i k_i}_{=:r})
\end{aligned}
$$

$\mathcal{B}$ thus returns $(\mathbf{E}, \sigma, (v, r))$, which makes it win the game EUF-CRO.

Together this shows that whenever $\mathcal{A}$ wins Game_5 then $\mathcal{B}$ wins EUF-CRO, that is

$$
\mathsf{Adv}^{\text{euf-cro}}_{\mathsf{Com},\mathsf{Sig},\mathcal{B}}(\lambda) = \mathsf{Adv}^{\text{Game}_5}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) \ . \tag{5.16}
$$

The theorem now follows from Eqs. (5.10) to (5.16). □

5.6 Transaction-indistinguishability

Theorem 5.12 (Transaction indistinguishability (Def. 5.8))**.** *Assume that* Com *is a homomorphic hiding commitment scheme,* Sig *a compatible signature scheme, and* Π *is a zero-knowledge proof system. Then the aggregate cash system* $\mathsf{MW}[\mathsf{Com}, \mathsf{Sig}, \Pi]$ *is transaction-indistinguishable. More precisely, for any* $v_{\max}$ *and any* PPT *adversary* $\mathcal{A}$ *which makes at most* $q_{\mathcal{A}}$ *queries to its oracle* TX*, there exist* PPT *adversaries* $\mathcal{B}_{zk}$ *and* $\mathcal{B}_{hid}$ *such that:*

$$
\mathsf{Adv}^{\text{tx-ind}}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) \leq \mathsf{Adv}^{\text{zk}}_{\Pi,\mathsf{R}_{v_{\max}},\mathcal{B}_{zk}}(\lambda) + q_{\mathcal{A}} \cdot \mathsf{Adv}^{\text{hid}}_{\mathsf{Com},\mathcal{B}_{hid}}(\lambda) \ .
$$

Adversary $\mathcal{B}^{\text{SIGN}'}(\mathsf{cp}, \mathsf{sp}, C^*)$

$\tilde{c} \leftarrow\!\$\, [h_{\mathcal{A}}]\,;\ \tilde{\imath} \leftarrow\!\$\, [n_{\mathcal{A}}]\,;\ c \coloneqq 0$
$\widetilde{C} \coloneqq \bot\,;\ \widetilde{\mathsf{ptx}} \coloneqq \bot\,;\ \overline{C} \coloneqq \bot$
$(\mathsf{crs}, \tau) \leftarrow \Pi.\mathsf{G}(\Gamma, v_{\max})$
$\Lambda \coloneqq \big(0, (\,), (\,), ((\,), (\,), \mathbf{e})\big)$
$\mathsf{Chg}, \mathsf{Hon}, \mathsf{Val}, \mathsf{Key}, \mathsf{Ptx} \coloneqq (\,)$
$\mathcal{A}^{\text{MINT},\text{SEND},\text{RECEIVE},\text{LEDGER}}((\mathsf{cp}, \mathsf{sp}, \mathsf{crs}), \Lambda)$
return $\bot$ // if not stopped earlier

MW.Coin$(\mathsf{pp}, \mathbf{v}, j)$

for $i = 1 \ldots |\mathbf{v}|$ **do**
  $c \coloneqq c + 1\,;\ k_i \leftarrow\!\$\, \mathcal{R}_{\mathsf{cp}}$
  if $c = \tilde{c}$ **or** $i = j$:
    $C_i \coloneqq C^* + \mathsf{Com.C}(\mathsf{cp}, v_i; k_i)$ // embed
    $\pi_i \leftarrow \Pi.\mathsf{Sim}(\mathsf{crs}, \tau, (\mathsf{cp}, C_i))$ // challenge
  else: $C_i \coloneqq \mathsf{Com.C}(\mathsf{cp}, v_i; k_i)$
    $\pi_i \leftarrow \Pi.\mathsf{P}(\mathsf{crs}, (\mathsf{cp}, C_i), (v_i, k_i))$
  if $c = \tilde{c}$: $\widetilde{C} \coloneqq C_i$
return $\big((C_i)_{i=1}^{|\mathbf{v}|}, (k_i)_{i=1}^{|\mathbf{v}|}, (\pi_i)_{i=1}^{|\mathbf{v}|}\big)$

MW.MkTx$(\mathsf{pp}, (\mathbf{C}, \mathbf{v}, \mathbf{k}), \hat{\mathbf{v}})$

if $\neg\, \mathsf{Cons}(\mathsf{pp}, \mathbf{C}, \mathbf{v}, \mathbf{k})$: **return** $\bot$
$s \coloneqq \sum \hat{\mathbf{v}} - \sum \mathbf{v}$
if $\mathbf{v} \,\|\, \hat{\mathbf{v}} \not\subseteq [0, v_{\max}]^*$ **or** $s < 0$
  return $\bot$
$(\hat{\mathbf{C}}, \hat{\mathbf{k}}, \hat{\pi}) \leftarrow \mathsf{MW.Coin}(\mathsf{pp}, \hat{\mathbf{v}})$
if $\widetilde{C} \in \hat{\mathbf{C}}$ // $\widetilde{C}$ created in this tx
  $\text{SIGN}'(\sum \hat{\mathbf{k}} - \sum \mathbf{k}, \mathbf{e})$
else : $\sigma \leftarrow \mathsf{Sig.S}(\mathsf{sp}, \sum \hat{\mathbf{k}} - \sum \mathbf{k}, \mathbf{e})$
$\mathsf{tx} \coloneqq \big(s, \mathbf{C}, \hat{\mathbf{C}}, (\hat{\pi}, \sum \hat{\mathbf{C}} - \sum \mathbf{C}, \sigma)\big)$
return $(\mathsf{tx}, \hat{\mathbf{k}})$

Procedure Finalize(tx)

$\mathsf{tx} = (s, \mathbf{C}, \hat{\mathbf{C}}, (\hat{\pi}, \mathbf{E}, \sigma))\,;\ \mathbf{C}' \coloneqq \mathbf{C} - (\widetilde{C})$
let π' be the proofs for $\mathbf{C}'$ in Λ
$(\mathbf{v}, \mathbf{k}) \coloneqq \Pi.\mathsf{Ext}(\mathsf{crs}, \tau, (\mathsf{cp}, \mathbf{C}'), \pi')$
$(\hat{\mathbf{v}}, \hat{\mathbf{k}}) \coloneqq \Pi.\mathsf{Ext}(\mathsf{crs}, \tau, (\mathsf{cp}, \hat{\mathbf{C}}), \hat{\pi})$
if $\neg\, \mathsf{R}_{v_{\max}}\big((\mathsf{cp}, \mathbf{C}' \,\|\, \hat{\mathbf{C}}), (\mathbf{v} \,\|\, \hat{\mathbf{v}}, \mathbf{k} \,\|\, \hat{\mathbf{k}})\big)$
  abort
return $(\mathbf{E}, \sigma, \sum \hat{\mathbf{v}} - \sum \mathbf{v} - s, \sum \hat{\mathbf{k}} - \sum \mathbf{k})$

MW.S$(\mathsf{pp}, (\mathbf{C}, \mathbf{v}, \mathbf{k}), \mathbf{v}')$

$\rho \coloneqq \sum \mathbf{v} - \sum \mathbf{v}'$
if $\neg\, \mathsf{Cons}(\mathsf{pp}, \mathbf{C}, \mathbf{v}, \mathbf{k})$: **return** $\bot$
if $\mathbf{v} \,\|\, \mathbf{v}' \,\|\, \rho \not\subseteq [0, v_{\max}]^*$: **return** $\bot$
if $\overline{C} \in \mathbf{C}$: **abort**
if $\widetilde{C} \in \mathbf{C}$ // ptx being created will be $\widetilde{\mathsf{ptx}}$
  if $|\mathbf{v}'| < \tilde{\imath}$: **abort**
  // embed challenge in $\tilde{\imath}$-th change coin:
  $(\hat{\mathbf{C}}, \hat{\mathbf{k}}, \hat{\pi}) \leftarrow \mathsf{MW.Coin}(\mathsf{pp}, \mathbf{v}' \,\|\, \rho, \tilde{\imath})$
  $\overline{C} \coloneqq \hat{\mathbf{C}}[\tilde{\imath}]$
else: $(\hat{\mathbf{C}}, \hat{\mathbf{k}}, \hat{\pi}) \leftarrow \mathsf{MW.Coin}(\mathsf{pp}, \mathbf{v}' \,\|\, \rho)$
$\sigma \leftarrow \mathsf{Sig.S}(\mathsf{sp}, \sum \hat{\mathbf{k}} - \sum \mathbf{k}, \mathbf{e})$
$\mathsf{tx} \coloneqq \big(0, \mathbf{C}, \hat{\mathbf{C}}, (\hat{\pi}, \sum \hat{\mathbf{C}} - \sum \mathbf{C}, \sigma)\big)$
return $\big(\mathsf{ptx} \coloneqq (\mathsf{tx}, \rho, \hat{k}_{|\mathbf{v}'|+1}), (\hat{k}_i)_{i=1}^{|\mathbf{v}'|}\big)$

Oracle LEDGER(tx)

$\Lambda' \leftarrow \mathsf{MW.L}(\mathsf{pp}, \Lambda, \mathsf{tx})$
if $\Lambda' = \bot$: **return** $\bot$ **else** : $\Lambda \coloneqq \Lambda'$
for all $\mathsf{ptx} \in \mathsf{Ptx}$ **do**
  if $\mathsf{ptx.chg} \subseteq \mathsf{tx.out}$
    $\mathsf{Ptx} \coloneqq \mathsf{Ptx} - (\mathsf{ptx})$
    $\mathsf{Hon} \coloneqq (\mathsf{Hon} - \mathsf{ptx.in}) \,\|\, \mathsf{ptx.chg}$
    $\mathsf{Chg} \coloneqq \mathsf{Chg} - \mathsf{ptx.chg}$
if $\mathsf{tx.in} \cap \mathsf{Chg} \neq (\,)$
  if $\widetilde{C} = \mathsf{tx.in}[j]$ with $j = \min\{i \,|\, \mathsf{tx.in}[i] \in \mathsf{Chg}\}$
    Finalize(tx)
  else : **abort** (I)
if $\mathsf{tx.in} \cap \mathsf{Hon} \neq (\,)$
  if $\widetilde{C} = \mathsf{tx.in}[j]$ with $j = \min\{i \,|\, \mathsf{tx.in}[i] \in \mathsf{Hon}\}$:
    $\wedge\ (\widetilde{\mathsf{ptx}} = \bot \vee \tilde{\imath} = \min\{i \,|\, \widetilde{\mathsf{ptx}}.\mathsf{chg}[i] \notin \mathsf{tx.out}\})$
      // $\overline{C} = \widetilde{\mathsf{ptx}}.\mathsf{chg}[\tilde{\imath}]$
    Finalize(tx)
  else : **abort** (II)
return Λ

Figure 5.12: Reduction $\mathcal{B}$ simulating Game_5 (only showing oracles that differ from Fig. 5.11).

Game $\text{HID-PRR}^b_{\mathsf{Com},\mathcal{A}}(1^\lambda)$

$\Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$
$\mathsf{cp} \leftarrow \mathsf{Com.G}(\Gamma)$
$(\mathbf{v}_0, \mathbf{v}_1, \mathsf{state}) \leftarrow \mathcal{A}(\mathsf{cp})$
if $|\mathbf{v}_0| \neq |\mathbf{v}_1|$ **or** $\sum \mathbf{v}_0 \neq \sum \mathbf{v}_1$
  return 0
$(\mathbf{C}, \mathbf{r}) \coloneqq \mathsf{Com.C}(\mathsf{cp}, \mathbf{v}_b)$
$b' \leftarrow \mathcal{A}(\mathbf{C}, \sum \mathbf{r}, \mathsf{state})$
return b'

Adversary $\mathcal{B}(\mathsf{cp})$

$(\mathbf{v}_0, \mathbf{v}_1, \mathsf{state}) \leftarrow \mathcal{A}(\mathsf{cp})$
if $|\mathbf{v}_0| \neq |\mathbf{v}_1|$ **or** $\sum \mathbf{v}_0 \neq \sum \mathbf{v}_1$
  return 0
for $i = 1 \ldots |\mathbf{v}_0| - 1$ **do**
// oracle query in game $\text{HID}_{\mathsf{Com}}$:
  $C_i \leftarrow \textsc{Commit}(v_0[i], v_1[i])$
$\mathbf{C}' \coloneqq (C_i)_{i=1}^{|\mathbf{v}_0|-1}$; $k \leftarrow\!\$\, \mathcal{R}_{\mathsf{cp}}$
$b' \leftarrow \mathcal{A}\big(\mathbf{C}' \parallel \big(\mathsf{Com.C}(\mathsf{cp}, \sum \mathbf{v}_0; k) - \sum \mathbf{C}'\big), k), \mathsf{state}\big)$
return b'

Figure 5.13: Game HID-PRR and adversary $\mathcal{B}$ for Lemma 5.14.

Before proving the theorem, we show a fact that will be useful for the proof. Consider commitment parameters cp for Com and let $v_b, v'_b \in \mathcal{V}_{\mathsf{cp}}$; then the following distributions are all equivalent:

$$\left[r, r' \leftarrow\!\$\, \mathcal{R}_{\mathsf{cp}} : C \coloneqq \mathsf{Com.C}(\mathsf{cp}, v_b; r),\ C' \coloneqq \mathsf{Com.C}(\mathsf{cp}, v'_b; r');\ k \coloneqq r + r'\right], \tag{5.17}$$

$$\left[r, r' \leftarrow\!\$\, \mathcal{R}_{\mathsf{cp}} : C \coloneqq \mathsf{Com.C}(\mathsf{cp}, v_b; r),\ C' \coloneqq \mathsf{Com.C}(\mathsf{cp}, v_b + v'_b; r + r') - C;\ k \coloneqq r + r'\right], \tag{5.18}$$

$$\left[r, k \leftarrow\!\$\, \mathcal{R}_{\mathsf{cp}} : C \coloneqq \mathsf{Com.C}(\mathsf{cp}, v_b; r),\ C' \coloneqq \mathsf{Com.C}(\mathsf{cp}, v_b + v'_b; k) - C,\ k\right]. \tag{5.19}$$

(The distribution in) (5.18) is equivalent to (the one in) (5.17) since Com is additively homomorphic; (5.19) is equivalent to (5.18) since $\mathcal{R}_{\mathsf{cp}}$ is a group and therefore $r + r'$ and k are equally distributed.

Now consider an adversary $\mathcal{A}$ that chooses $v_0, v'_0, v_1, v'_1 \in \mathcal{V}_{\mathsf{cp}}$ such that $v_0 + v'_0 = v_1 + v'_1$ and receives a tuple (C, C', k) as defined in (5.17) for a random $b \leftarrow\!\$\, \{0, 1\}$ and $\mathcal{A}$ has to guess b. Then if Com is hiding, $\mathcal{A}$'s advantage will be negligible; intuitively, this is because (5.17) is distributed as (5.19) and in the latter the only thing depending on b is C (since $v_0 + v'_0 = v_1 + v'_1$). More formally, one can construct an adversary $\mathcal{B}$ for the game $\text{HID}^b_{\mathsf{Com},\mathcal{A}}(1^\lambda)$ game which queries its challenge oracle on (v_0, v_1) to get C and simulates distribution (5.17) that $\mathcal{A}$ expects using (5.19).

We generalize this indistinguishability notion to vectors of values of length more than two as follows:

Definition 5.13 (HID-PRR)**.** *Let game* HID-PRR *be as defined in Fig. 5.13. A commitment scheme* Com *is* hiding under partially revealed randomness *if for any* PPT *adversary* $\mathcal{A}$*:*

$$\mathsf{Adv}^{\text{hid-prr}}_{\mathsf{Com},\mathcal{A}}(\lambda) \coloneqq \left|\Pr\left[\text{HID-PRR}^0_{\mathsf{Com},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\text{HID-PRR}^1_{\mathsf{Com},\mathcal{A}}(\lambda) = 1\right]\right| = \mathsf{negl}(\lambda).$$

The following is proved by generalizing reduction $\mathcal{B}$ sketched above.

Lemma 5.14. *Any hiding homomorphic commitment scheme* Com *is also* HID-PRR*. More precisely, for any* PPT *adversary* $\mathcal{A}$*, there exists a* PPT *adversary* $\mathcal{B}$ *such that:*

$$\mathsf{Adv}^{\text{hid-prr}}_{\mathsf{Com},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\text{hid}}_{\mathsf{Com},\mathcal{B}}(\lambda).$$

Proof. Fix values $\mathbf{v}_b$ and let $\mathbf{v}'_b$ denote the first $|\mathbf{v}_b| - 1$ components of $\mathbf{v}_b$. Then, as above, the following distributions can be shown to be the same:

$$\left[\mathbf{r} \leftarrow\!\$\, (\mathcal{R}_{\mathsf{cp}})^{|\mathbf{v}_b|} : \mathbf{C} \coloneqq \mathsf{Com.C}(\mathsf{cp}, \mathbf{v}_b; \mathbf{r}),\ k \coloneqq \sum \mathbf{r}\right],$$

$$\left[\mathbf{r}' \leftarrow\!\$\, (\mathcal{R}_{\mathsf{cp}})^{|\mathbf{v}_b|-1}, k \leftarrow\!\$\, \mathcal{R}_{\mathsf{cp}} : (\mathbf{C}' \coloneqq \mathsf{Com.C}(\mathsf{cp}, \mathbf{v}'_b; \mathbf{r}')) \parallel \left(\mathsf{Com.C}(\mathsf{cp}, \textstyle\sum \mathbf{v}_b; k) - \sum \mathbf{C}'\right),\ k\right].$$

$\underline{\text{IND-TX}^{b}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) \text{ Game}_1}$

$\Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$
$\mathsf{cp} \leftarrow \mathsf{Com.G}(\Gamma)$
$\mathsf{sp} \leftarrow \mathsf{Sig.G}(\Gamma)$
$(\mathsf{crs}, \tau) \leftarrow \Pi.\mathsf{G}(\Gamma, v_{\max})$
$b' \leftarrow \mathcal{A}^{\text{Tx}}(\mathsf{pp})$
return b'

$\underline{\mathsf{MW.Coin}((\mathsf{cp}, \mathsf{sp}, \mathsf{crs}), \mathbf{v})}$

for $i = 1 \ldots |\mathbf{v}|$ **do**
  $k_i \leftarrow\!\!\$\ \mathcal{R}_{\mathsf{cp}}$
  $C_i := \mathsf{Com.C}(\mathsf{cp}, v_i; k_i)$
  $\pi_i \leftarrow \Pi.\mathsf{P}(\mathsf{crs}, (\mathsf{cp}, C_i), (v_i, k_i))$
  $\boxed{\pi_i \leftarrow \Pi.\mathsf{Sim}(\mathsf{crs}, \tau, (\mathsf{cp}, C_i))}$
return $\big((C_i)_{i=1}^{|\mathbf{v}|}, (k_i)_{i=1}^{|\mathbf{v}|}, (\pi_i)_{i=1}^{|\mathbf{v}|}\big)$

$\underline{\mathsf{MW.MkTx}(\mathsf{pp}, (\mathbf{C}, \mathbf{v}, \mathbf{k}), \hat{\mathbf{v}})}$

$s := \sum \hat{\mathbf{v}} - \sum \mathbf{v}$
$(\hat{\mathbf{C}}, \hat{\mathbf{k}}, \hat{\pi}) \leftarrow \mathsf{MW.Coin}(\mathsf{pp}, \hat{\mathbf{v}})$
$E := \sum \hat{\mathbf{C}} - \sum \mathbf{C} - \mathsf{Com.C}(\mathsf{cp}, s; 0)$
$\sigma \leftarrow \mathsf{Sig.S}(\mathsf{sp}, \sum \hat{\mathbf{k}} - \sum \mathbf{k}, \mathbf{e})$
return $(\mathsf{tx} := (s, \mathbf{C}, \hat{\mathbf{C}}, (\hat{\pi}, E, \sigma)), \hat{\mathbf{k}})$

$\underline{\text{Oracle } \text{Tx}((\mathbf{v}_0, \mathbf{v}'_0, \mathbf{v}''_0), (\mathbf{v}_1, \mathbf{v}'_1, \mathbf{v}''_1))}$

if not $(\mathbf{v}_0, \mathbf{v}'_0, \mathbf{v}''_0, \mathbf{v}_1, \mathbf{v}'_1, \mathbf{v}''_1 \in [0, v_{\max}]^*)$
  return $\perp$
if $|\mathbf{v}_0| \neq |\mathbf{v}_1|$ **or** $|\mathbf{v}'_0| + |\mathbf{v}''_0| \neq |\mathbf{v}'_1| + |\mathbf{v}''_1|$
  return $\perp$
if $\sum \mathbf{v}_0 \neq \sum(\mathbf{v}'_0 \,\|\, \mathbf{v}''_0)$ **or** $\sum \mathbf{v}_1 \neq \sum(\mathbf{v}'_1 \,\|\, \mathbf{v}''_1)$
  return $\perp$
// $(\mathsf{tx}, \mathbf{k}) \leftarrow \mathsf{M}(\mathsf{pp}, \mathbf{v}_b)$
$(\mathsf{tx}, \mathbf{k}) \leftarrow \mathsf{MW.MkTx}(\mathsf{pp}, ((\,), (\,), (\,)), \mathbf{v}_b)$
$(\sum \mathbf{v}_b, (\,), \mathbf{C}, K) := \mathsf{tx}$
// $(\mathsf{ptx}, \mathbf{k}') \leftarrow \mathsf{S}(\mathsf{pp}, (\mathsf{tx.out}, \mathbf{v}_b, \mathbf{k}), \mathbf{v}'_b)$
$\rho := \sum \mathbf{v}_b - \sum \mathbf{v}'_b$
$(\mathsf{tx}', \mathbf{k}' \,\|\, k^*) \leftarrow \mathsf{MW.MkTx}(\mathsf{pp}, (\mathbf{C}, \mathbf{v}_b, \mathbf{k}), \mathbf{v}'_b \,\|\, \rho)$
$\big(0, \mathbf{C}, \mathbf{C}' \,\|\, C^*, (\pi' \,\|\, \pi^*, E', \sigma')\big) := \mathsf{tx}'$
  // where $E' = \sum \mathbf{C}' + C^* - \sum \mathbf{C} - 0$
  // and $\sigma' \leftarrow \mathsf{S}\big(\mathsf{sp}, \sum \mathbf{k}' + k^* - \sum \mathbf{k}, \mathbf{e}\big)$
// $(\mathsf{tx}^*, \mathbf{k}'') \leftarrow \mathsf{R}(\mathsf{pp}, \mathsf{ptx}, \mathbf{v}''_b)$ with $\mathsf{ptx} := (\mathsf{tx}', \rho, k^*)$
  // note that $\rho = \sum \mathbf{v}''_b$
$(\mathsf{tx}'', \mathbf{k}'') \leftarrow \mathsf{MW.MkTx}(\mathsf{pp}, (C^*, \rho, k^*), \mathbf{v}''_b)$
$\big(0, (C^*), \mathbf{C}'', (\pi'', E'', \sigma'')\big) := \mathsf{tx}''$
  // where $E'' = \sum \mathbf{C}'' - C^* - 0$
  // and $\sigma'' \leftarrow \mathsf{S}\big(\mathsf{sp}, \sum \mathbf{k}'' - k^*, \mathbf{e}\big)$
$\mathsf{tx}^* \leftarrow \mathsf{MW.A}(\mathsf{pp}, \mathsf{tx}', \mathsf{tx}'')$
  // if $\mathsf{tx}^* \neq \perp$:
$\big(0, \mathbf{C}, \mathbf{C}' \,\|\, \mathbf{C}'', (\pi' \,\|\, \pi'', (E', E''), \sigma^*)\big) := \mathsf{tx}^*$
  // where $\sigma^* \leftarrow \mathsf{Sig.A}\big(\mathsf{sp}, ((E'), \sigma'), ((E''), \sigma'')\big)$
return tx^*

Figure 5.14: Transaction-indistinguishability game for MW and hybrid game.

Let $\mathcal{A}$ be an adversary for game HID-PRR$_{\mathsf{Com}}$. We construct an adversary $\mathcal{B}$ for game HID$_{\mathsf{Com}}$ as shown in Fig. 5.13. By the observation above, game HID-PRR$_{\mathsf{Com}}$ is perfectly simulated by $\mathcal{B}$. Hence, $\mathcal{B}$ wins HID$_{\mathsf{Com}}$ with the same advantage as $\mathcal{A}$ distinguishes $b = 0$ from $b = 1$ in game HID-PRR$_{\mathsf{Com}}$. □

Before proving Theorem 5.12, we state another simple fact.

Lemma 5.15. *Let* Com *be an (additively) homomorphic commitment scheme, parameters* cp *be output by* Com.G*,* $v \in \mathcal{V}_{\mathsf{cp}}$ *and* $r \in \mathcal{R}_{\mathsf{cp}}$*. Then* $\mathsf{Com.C}(\mathsf{cp}, -v; -r) = -\mathsf{Com.C}(\mathsf{cp}, v; r)$*.*

Proof. We have $\mathsf{Com.C}(\mathsf{cp}, 0; 0) = 0$, since $\mathsf{Com.C}(\mathsf{cp}, v; r) = \mathsf{Com.C}(\mathsf{cp}, 0{+}v; 0{+}r) = \mathsf{Com.C}(\mathsf{cp}, 0; 0){+}$ $\mathsf{Com.C}(\mathsf{cp}, v; r)$. This implies $0 = \mathsf{Com.C}(\mathsf{cp}, v - v; r - r) = \mathsf{Com.C}(\mathsf{cp}, v; r) + \mathsf{Com.C}(\mathsf{cp}, -v; -r)$, which shows the statement. □

Proof of Theorem 5.12. We start with instantiating Cash in Fig. 5.6 with MW and write out game TX-IND$_{\mathsf{MW}}$ in Fig. 5.14, where the boxes should be ignored. (We have simplified the description by omitting checks for inputs that are created correctly by the experiment.) We next define a game Game_1 (Fig. 5.14, including the boxes) where all range proofs are simulated. It is straightforward to construct an adversary $\mathcal{B}_{zk}$ so that

$$\mathsf{Adv}^{\mathrm{Game}_1}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) \geq \mathsf{Adv}^{\text{tx-ind}}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) - \mathsf{Adv}^{\mathrm{zk}}_{\Pi,\mathsf{R}_{v_{\max}},\mathcal{B}_{zk}}(\lambda) \ .$$

By Lemma 5.14, in order to prove the theorem, it suffices to construct $\mathcal{B}$ such that

$$\mathsf{Adv}^{\text{hid-prr}}_{\mathsf{Com},\mathcal{B}}(\lambda) \geq \tfrac{1}{q_{\mathcal{A}}} \cdot \mathsf{Adv}^{\mathrm{Game}_1}_{\mathsf{MW},\mathcal{A}}(\lambda, v_{\max}) \ . \tag{5.20}$$

Consider a Tx-oracle query $(\mathbf{v}_0, \mathbf{v}'_0, \mathbf{v}''_0), (\mathbf{v}_1, \mathbf{v}'_1, \mathbf{v}''_1)$ with $\mathbf{v}_0, \mathbf{v}'_0, \mathbf{v}''_0, \mathbf{v}_1, \mathbf{v}'_1, \mathbf{v}''_1 \in [0, v_{\max}]^*$ and

$$\begin{aligned} |\mathbf{v}_0| &= |\mathbf{v}_1| \\ |\mathbf{v}'_0| + |\mathbf{v}''_0| &= |\mathbf{v}'_1| + |\mathbf{v}''_1| \qquad \textstyle\sum \mathbf{v}'_0 + \sum \mathbf{v}''_0 - \sum \mathbf{v}_0 = 0 = \sum \mathbf{v}'_1 + \sum \mathbf{v}''_1 - \sum \mathbf{v}_1 \ . \end{aligned} \tag{5.21}$$

We will show that the response of Tx is independent of b. (If one of the conditions in (5.21) does not hold, then Tx returns $\perp$, independently of b). By Fig. 5.14, the oracle reply is of the form

$$\mathsf{tx}^* = \Big(0, \mathbf{C}, \mathbf{C}' \,\|\, \mathbf{C}'', \Big(\pi' \,\|\, \pi'', (E', E''), \mathsf{Sig.A}\big(\mathsf{sp}, ((E'), \sigma'), ((E''), \sigma'')\big)\Big)\Big) \quad \text{with} \tag{5.22}$$

$$\begin{aligned} \sigma' &\leftarrow \mathsf{S}\big(\mathsf{sp}, \textstyle\sum \mathbf{k}' + k^* - \sum \mathbf{k}, \mathbf{e}\big) \\ \sigma'' &\leftarrow \mathsf{S}\big(\mathsf{sp}, \textstyle\sum \mathbf{k}'' - k^*, \mathbf{e}\big) \\ E' = \textstyle\sum \mathbf{C}' + C^* - \sum \mathbf{C} &= \mathsf{Com.C}\big(\mathsf{cp},\ \textstyle\sum \mathbf{v}'_b + \rho - \sum \mathbf{v}_b;\ \sum \mathbf{k}' + k^* - \sum \mathbf{k}\big) \\ &= \mathsf{Com.C}\big(\mathsf{cp},\ 0;\ \textstyle\sum \mathbf{k}' + k^* - \sum \mathbf{k}\big) \qquad (5.23) \\ E'' = \textstyle\sum \mathbf{C}'' - C^* &= \mathsf{Com.C}\big(\mathsf{cp},\ \textstyle\sum \mathbf{v}''_b - \rho;\ \sum \mathbf{k}'' - k^*\big) = \mathsf{Com.C}\big(\mathsf{cp},\ 0;\ \textstyle\sum \mathbf{k}'' - k^*\big) \ , \qquad (5.24) \end{aligned}$$

where the last equations in (5.23) and (5.24) follow since Com is homomorphic and $\sum \mathbf{v}_b - \sum \mathbf{v}'_b = \rho = \sum \mathbf{v}''_b$, by the definition of ρ and (5.21).

On the other hand, for vectors $\mathbf{v}_b, \mathbf{v}'_b, \mathbf{v}''_b$, for which (5.21) holds, we have that the distribution

$$\Big[\mathbf{k} \,\|\, \mathbf{k}' \,\|\, \mathbf{k}'' \leftarrow\!\$\, \mathcal{R}_{\mathsf{cp}}^{|\mathbf{v}_0|+|\mathbf{v}'_0|+|\mathbf{v}''_0|} : \mathbf{C} \,\|\, \mathbf{C}' \,\|\, \mathbf{C}'',\ \overline{k} \coloneqq \textstyle\sum \mathbf{k}' + \sum \mathbf{k}'' - \sum \mathbf{k}\Big] \tag{5.25}$$

with $\mathbf{C} \coloneqq \mathsf{C}(\mathsf{cp}, \mathbf{v}_b, \mathbf{k})$, $\mathbf{C}' \coloneqq \mathsf{C}(\mathsf{cp}, \mathbf{v}'_b, \mathbf{k}')$ and $\mathbf{C}'' \coloneqq \mathsf{C}(\mathsf{cp}, \mathbf{v}''_b, \mathbf{k}'')$ are indistinguishable for $b = 0$ or $b = 1$. This follows by applying Lemma 5.14 to vectors $-\mathbf{v}_b \,\|\, \mathbf{v}'_b \,\|\, \mathbf{v}''_b$, for $b = 0, 1$, and then using Lemma 5.15. From (5.25) we get that

$$\Big[\mathbf{k} \,\|\, \mathbf{k}' \,\|\, \mathbf{k}'' \,\|\, (k^*) \leftarrow\!\$\, \mathcal{R}_{\mathsf{cp}}^{|\mathbf{v}_0|+|\mathbf{v}'_0|+|\mathbf{v}''_0|+1} : \mathbf{C} \,\|\, \mathbf{C}' \,\|\, \mathbf{C}'',\ r' \coloneqq \textstyle\sum \mathbf{k}' + k^* - \sum \mathbf{k},\ r'' \coloneqq \sum \mathbf{k}'' - k^*\Big] \tag{5.26}$$

is also indistinguishable for $b = 0$ and $b = 1$: We could construct a reduction $\mathcal{B}_{(5.26)}$ that, given $(\mathbf{C} \,\|\, \mathbf{C}' \,\|\, \mathbf{C}'', \overline{k})$ distributed as in (5.25), samples $r \leftarrow \mathcal{R}_{\mathsf{cp}}$ and runs a distinguisher for (5.26) on $(\mathbf{C} \,\|\, \mathbf{C}' \,\|\, \mathbf{C}'', r, \overline{k} - r)$; the latter is distributed correctly, since r' and r'' are uniform conditioned on $r' + r'' = \overline{k}$.

Since oracle Tx does not reveal C^* and thus k^* is perfectly hidden, (5.26) implies that tx^* in (5.22) is also indistinguishable for $b = 0$ or $b = 1$: we can construct an adversary $\mathcal{B}_{(5.22)}$ which, given an output of the form (5.26), computes a tuple of the form (5.22) by setting:

$$\begin{aligned} \pi' &\leftarrow \Pi.\mathsf{Sim}(\mathsf{crs}, \tau, (\mathsf{cp}, \mathbf{C}')) \qquad & \sigma' &\leftarrow \mathsf{Sig.S}\big(\mathsf{sp}, r', \mathbf{e}\big) \qquad & E' &= \mathsf{Com.C}\big(\mathsf{cp}, 0; r'\big) \\ \pi'' &\leftarrow \Pi.\mathsf{Sim}(\mathsf{crs}, \tau, (\mathsf{cp}, \mathbf{C}'')) \qquad & \sigma'' &\leftarrow \mathsf{Sig.S}\big(\mathsf{sp}, r'', \mathbf{e}\big) \qquad & E'' &= \mathsf{Com.C}\big(\mathsf{cp}, 0; r''\big) \ , \end{aligned}$$

where we additionally used that $|\mathbf{C}_0| = |\mathbf{C}_1|$ and $|(\mathbf{C}'_0 \,\|\, \mathbf{C}''_0)| = |(\mathbf{C}'_1 \,\|\, \mathbf{C}''_1)|$, as implied by (5.21). All oracle replies are thus distinguishable with advantage at most $\mathsf{Adv}^{\text{hid-prr}}_{\mathsf{Com},\mathcal{B}}(\lambda)$, where $\mathcal{B}$ combines adversaries $\mathcal{B}_{(5.26)}$ and $\mathcal{B}_{(5.22)}$, defined above. This shows (5.20) and thus the theorem. □

Game $\mathrm{DL}_{\mathsf{GrGen},\mathcal{A}}(\lambda)$	Game $\mathrm{CDH}_{\mathsf{GrGen}',\mathcal{A}}(\lambda)$
$\Gamma := (p, \mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^\lambda)$	$\Gamma' := (p, \mathbb{G}, \mathbb{G}_T, e, G) \leftarrow \mathsf{GrGen}'(1^\lambda)$
$x \leftarrow\!\$\, \mathbb{Z}_p\,;\ X := xG$	$x \leftarrow\!\$\, \mathbb{Z}_p\,; y \leftarrow\!\$\, \mathbb{Z}_p$
$x' \leftarrow \mathcal{A}(\Gamma, X)$	$Z \leftarrow \mathcal{A}(\Gamma, xG, yG)$
return $x' = x$	**return** $Z = xyG$

Figure 5.15: The Discrete Logarithm and Computational Diffie-Hellman games.

5.7 Instantiations

A group description is a tuple $\Gamma = (p, \mathbb{G}, G)$ where p is an odd prime of length λ, $\mathbb{G}$ is an additive abelian group of prime order p, and G is a generator of $\mathbb{G}$. A bilinear group description is a tuple $\Gamma' = (p, \mathbb{G}, \mathbb{G}_T, e, G)$ where p is an odd prime of length λ, $\mathbb{G}$ and $\mathbb{G}_T$ are groups of order p (we denote $\mathbb{G}_T$ multiplicatively), G is a generator of $\mathbb{G}$ and e is an efficiently computable non-degenerate bilinear map $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ (i.e., the map e is such that for all $U, V \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, $e(aU, bV) = e(U, V)^{ab}$, and $e(G, G)$ is a generator of $\mathbb{G}_T$). We assume the existence of a PPT algorithm GrGen (GrGen') that, given as input the security parameter in unary 1^λ, outputs a group description Γ (a bilinear group description Γ').

For groups generated by GrGen we will make the assumption that discrete logarithms (DL) are hard to compute, while for bilinear groups generated by GrGen' we will make the computational Diffie-Hellman (CDH) assumption. They state that the advantages:

$$\mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GrGen},\mathcal{A}_{\mathrm{dl}}}(\lambda) := \Pr\left[\mathrm{DL}_{\mathsf{GrGen},\mathcal{A}_{\mathrm{dl}}}(\lambda) = 1\right]$$
$$\mathsf{Adv}^{\mathrm{cdh}}_{\mathsf{GrGen}',\mathcal{A}_{\mathrm{cdh}}}(\lambda) := \Pr\left[\mathrm{CDH}_{\mathsf{GrGen}',\mathcal{A}_{\mathrm{cdh}}}(\lambda) = 1\right]$$

and are negligible in λ for all PPT adversaries $\mathcal{A}_{\mathrm{dl}}$ and $\mathcal{A}_{\mathrm{cdh}}$, where games DL and CDH are specified in Fig. 5.15.

For the Pedersen-Schnorr (Pedersen-BLS) instantiation, the main setup algorithm GrGen consists of a (bilinear) group generation algorithm GrGen (GrGen').

Pedersen Commitments. The homomorphic commitment scheme proposed by Pedersen [Ped92], denoted PDS, is defined as:

$\mathsf{PDS.G}(\Gamma)$	$\mathsf{PDS.C}(\mathsf{cp}, v, r)$
$(p, \mathbb{G}, G) := \Gamma\,;\ H \leftarrow\!\$\, \mathbb{G}$	$((p, \mathbb{G}, G), H) := \mathsf{cp}$
return $\mathsf{cp} := (\Gamma, H)$	**return** $C := vH + rG$

A commitment C is opened by providing the value v and the randomness r. Pedersen commitments are computationally binding under the DL assumption and perfectly hiding. Since $\mathsf{PDS.C}(\mathsf{cp}, v_0; r_0) + \mathsf{PDS.C}(\mathsf{cp}, v_1; r_1) = \mathsf{PDS.C}(\mathsf{cp}, v_0 + v_1; r_0 + r_1)$, Pedersen commitments are additively homomorphic. PDS translates immediately to the case of a bilinear group description Γ'.

Schnorr Signatures. We recall the Schnorr signature scheme [Sch91] in Fig. 5.16. Note that we use the key-prefixed variant of the scheme, where the public key is hashed together with the commitment and the message. This corresponds to the *strong* Fiat-Shamir transform as defined in [BPW12], which ensures extractability in situations where the adversary can select public keys

$\underline{\mathsf{SCH.G}(\Gamma)}$

$(p, \mathbb{G}, G) \coloneqq \Gamma$
Select $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_p$
return $\mathsf{sp} \coloneqq (\Gamma, \mathcal{H})$

$\underline{\mathsf{SCH.K}(\mathsf{sp})}$

$((p, \mathbb{G}, G), \mathcal{H}) \coloneqq \mathsf{sp}$
$x \leftarrow\!\!\$\, \mathbb{Z}_p;\ X \coloneqq xG$
$\mathsf{sk} \coloneqq x;\ \mathsf{pk} \coloneqq X$
return $(\mathsf{sk}, \mathsf{pk})$

$\underline{\mathsf{SCH.A}(\mathsf{sp}, (\mathbf{L}_0, \boldsymbol{\sigma}_0), (\mathbf{L}_1, \boldsymbol{\sigma}_1))}$

return $\boldsymbol{\sigma}_0 \,\|\, \boldsymbol{\sigma}_1$

$\underline{\mathsf{SCH.S}(\mathsf{sp}, \mathsf{sk}, m)}$

$((p, \mathbb{G}, G), \mathcal{H}) \coloneqq \mathsf{sp}$
$x \coloneqq \mathsf{sk};\ X \coloneqq xG$
$r \leftarrow\!\!\$\, \mathbb{Z}_p;\ R \coloneqq rG$
$c \coloneqq \mathcal{H}(X, R, m);\ s \coloneqq r + cx$
return $\sigma \coloneqq (R, s)$

$\underline{\mathsf{SCH.V}(\mathsf{sp}, \mathbf{L}, \sigma)}$

$((p, \mathbb{G}, G), \mathcal{H}) \coloneqq \mathsf{sp}$
$((X_i, m_i))_{i=1}^n \coloneqq \mathbf{L}$
$((R_i, s_i))_{i=1}^n \coloneqq \boldsymbol{\sigma}$
for i **in** $[\![1, n]\!]$ **do** $c_i \coloneqq \mathcal{H}(X_i, R_i, m_i)$
return $\bigwedge_{i=1}^n (s_i G = R_i + c_i X_i)$

Figure 5.16: The Schnorr aggregate signature scheme.

adaptively, which is the case in the EUF-NZO and EUF-CRO security games. Note that no non-interactive aggregation procedure is known for Schnorr signatures other than trivially concatenating individual signatures.

Our security proofs for the Pedersen-Schnorr pair are in the random oracle model and make use of the standard rewinding technique of Pointcheval and Stern [PS00] for extracting discrete logarithms from a successful adversary. This requires some particular care since in both the EUF-NZO and the EUF-CRO games, the adversary can output multiple signatures for distinct public keys for which the reduction must extract discrete logarithms. Fortunately, a generalized forking lemma by Bagherzandi, Cheon, and Jarecki [BCJ08] shows that for Schnorr signatures, one can perform multiple extractions efficiently. Equipped with it, we can prove the following two lemmas, whose proofs can be found in the full version of the article [FOS19].

Lemma 5.16. *The pair* $(\mathsf{PDS}, \mathsf{SCH})$ *is EUF-NZO-secure in the random oracle model under the DL assumption. More precisely, for any p.p.t. adversary* $\mathcal{A}$ *making at most* q_h *random oracle queries and returning a forgery for a list of size at most* N*, there exists a* PPT *adversary* $\mathcal{B}$ *running in time at most* $8N^2 q_h/\delta_{\mathcal{A}} \cdot \ln(8N/\delta_{\mathcal{A}}) \cdot t_{\mathcal{A}}$*, where* $\delta_{\mathcal{A}} = \mathsf{Adv}^{\text{euf-nzo}}_{\mathsf{PDS},\mathsf{SCH},\mathcal{A}}(\lambda)$ *and* $t_{\mathcal{A}}$ *is the running time of* $\mathcal{A}$*, such that:*

$$\mathsf{Adv}^{\text{euf-nzo}}_{\mathsf{PDS},\mathsf{SCH},\mathcal{A}}(\lambda) \leq 8\ \mathsf{Adv}^{\text{dl}}_{\mathsf{GrGen},\mathcal{B}}(\lambda).$$

Lemma 5.17. *The pair* $(\mathsf{PDS}, \mathsf{SCH})$ *is EUF-CRO-secure in the random oracle model under the DL assumption. More precisely, for any p.p.t. adversary* $\mathcal{A}$ *making at most* q_h *random oracle queries and* q_s *signature queries, returning a forgery for a list of size at most* N*, and such that* $\delta_{\mathcal{A}} = \mathsf{Adv}^{\text{euf-cro}}_{\mathsf{PDS},\mathsf{SCH},\mathcal{A}}(\lambda) \geq 2q_s/p$*, there exists a* PPT *adversary* $\mathcal{B}$ *running in time at most* $16N^2(q_h + q_s)/\delta_{\mathcal{A}} \cdot \ln(16N/\delta_{\mathcal{A}}) \cdot t_{\mathcal{A}}$*, where* $t_{\mathcal{A}}$ *is the running time of* $\mathcal{A}$*, such that:*

$$\mathsf{Adv}^{\text{euf-nzo}}_{\mathsf{PDS},\mathsf{SCH},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\text{dl}}_{\mathsf{GrGen},\mathcal{B}}(\lambda) + \frac{q_s + 8}{p}.$$

Corollary 5.18. $\mathsf{MW}[\mathsf{PDS}, \mathsf{SCH}, \Pi]$ *with* Π *zero-knowledge and simulation-extractable is inflation-resistant and theft-resistant in the random oracle model under the DL assumption.*

$\underline{\mathsf{BLS.G}(\Gamma')}$

$(p, \mathbb{G}, \mathbb{G}_T, e, G) \coloneqq \Gamma'$
Select $\mathcal{H} : \{0,1\}^* \to \mathbb{G}$
return $\mathsf{sp} \coloneqq (\Gamma', \mathcal{H})$

$\underline{\mathsf{BLS.K}(\mathsf{sp})}$

$((p, \mathbb{G}, \mathbb{G}_T, e, G), \mathcal{H}) \coloneqq \mathsf{sp}$
$x \leftarrow\!\!\$\, \mathbb{Z}_p;\ X \coloneqq xG$
$\mathsf{sk} \coloneqq x;\ \mathsf{pk} \coloneqq X$
return $(\mathsf{sk}, \mathsf{pk})$

$\underline{\mathsf{BLS.A}(\mathsf{sp}, (\mathbf{L}_0, \sigma_0), (\mathbf{L}_1, \sigma_1))}$

return $\sigma_0 + \sigma_1$

$\underline{\mathsf{BLS.S}(\mathsf{sp}, \mathsf{sk}, m)}$

$((p, \mathbb{G}, \mathbb{G}_T, e, G), \mathcal{H}) \coloneqq \mathsf{sp}$
$x \coloneqq \mathsf{sk};\ X \coloneqq xG$
$Q \coloneqq \mathcal{H}(X, m)$
return $\sigma \coloneqq xQ$

$\underline{\mathsf{BLS.V}(\mathsf{sp}, \mathbf{L}, \sigma)}$

$((p, \mathbb{G}, \mathbb{G}_T, e, G), \mathcal{H}) \coloneqq \mathsf{sp}$
$((X_i, m_i))_{i=1}^n \coloneqq \mathbf{L}$
return $e(\sigma, G) = \prod_{i=1}^n e(X_i, \mathcal{H}(X_i, m_i))$

Figure 5.17: The BLS aggregate signature scheme.

BLS Signatures. The Boneh–Lynn–Shacham (BLS) signature scheme [BLS01] is a simple deterministic signature scheme based on pairings. It is defined in Fig. 5.17. We consider the key-prefixed variant of the scheme (i.e., the public key is hashed together with the message) which allows to securely aggregate signatures on the same message [BGLS03, BNN07]. EUF-CMA-security can be proved in the random oracle model under the CDH assumption.

The security proofs for the Pedersen-BLS pair are also in the random oracle model but do not use rewinding. They are reminiscent of the proof of [BGLS03, Theorem 3.2] and can be found in the full version [FOS19].

Lemma 5.19. *The pair* $(\mathsf{PDS}, \mathsf{BLS})$ *is EUF-NZO-secure in the random oracle model under the CDH assumption. More precisely, for any* PPT *adversary* $\mathcal{A}$ *making at most* q_h *random oracle queries and returning a forgery for a list of size at most* N*, there exists a* PPT *adversary* $\mathcal{B}$ *running in time at most* $t_{\mathcal{A}} + (q_h + N + 2)t_M$*, where* $t_{\mathcal{A}}$ *is the running time of* $\mathcal{A}$ *and* t_M *is the time of a scalar multiplication in* G*, such that:*

$$\mathsf{Adv}^{\mathrm{cdh}}_{\mathsf{GrGen},\mathcal{B}}(\lambda) = \mathsf{Adv}^{\text{euf-nzo}}_{\mathsf{PDS},\mathsf{BLS},\mathcal{A}}(\lambda).$$

Lemma 5.20. *The pair* $(\mathsf{PDS}, \mathsf{BLS})$ *is EUF-CRO-secure in the random oracle model under the CDH assumption. More precisely, for any* PPT*. adversary* A *making at most* q_h *random oracle queries and* $q_s = O(1)$ *signature queries and returning a forgery for a list of size at most* N*, there exists a* PPT *adversary* $\mathcal{B}$ *running in time at most* $t_{\mathcal{A}} + (2q_h + 3q_s + N + 2)t_M$*, where* $t_{\mathcal{A}}$ *is the running time of* $\mathcal{A}$ *and* t_M *is the time of a scalar multiplication in* $\mathbb{G}$*, such that:*

$$\mathsf{Adv}^{\mathrm{cdh}}_{\mathsf{GrGen},\mathcal{B}}(\lambda) \geq \frac{1}{4 \cdot (2N)^{q_s}} \cdot \mathsf{Adv}^{\text{euf-cro}}_{\mathsf{PDS},\mathsf{BLS},\mathcal{A}}(\lambda).$$

Corollary 5.21. $\mathsf{MW}[\mathsf{PDS}, \mathsf{BLS}, \Pi]$ *with* Π *zero-knowledge and simulation-extractable is inflation-resistant and theft-resistant in the random oracle model under the CDH assumption.*

Bibliography

[ABL+17] Divesh Aggarwal, Gavin K Brennen, Troy Lee, Miklos Santha, and Marco Tomamichel. Quantum attacks on bitcoin, and how to protect against them. *arXiv preprint arXiv:1710.10377*, 2017.

[ACFP14] Martin R. Albrecht, Carlos Cid, Jean-Charles Faugère, and Ludovic Perret. Algebraic algorithms for LWE. Cryptology ePrint Archive, Report 2014/1018, 2014. http://eprint.iacr.org/2014/1018.

[Adi08] Ben Adida. Helios: Web-based open-audit voting. In Paul C. van Oorschot, editor, *USENIX Security 2008*, pages 335–348. USENIX Association, July / August 2008.

[AFG14] Martin R. Albrecht, Robert Fitzpatrick, and Florian Göpfert. On the efficacy of solving LWE by reduction to unique-SVP. In Hyang-Sook Lee and Dong-Guk Han, editors, *ICISC 13*, volume 8565 of *LNCS*, pages 293–310. Springer, Heidelberg, November 2014.

[AG11] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In Luca Aceto, Monika Henzinger, and Jiri Sgall, editors, *ICALP 2011, Part I*, volume 6755 of *LNCS*, pages 403–415. Springer, Heidelberg, July 2011.

[AJL+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, Heidelberg, April 2012.

[AKR+13] Elli Androulaki, Ghassan Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. Evaluating user privacy in Bitcoin. In Ahmad-Reza Sadeghi, editor, *FC 2013*, volume 7859 of *LNCS*, pages 34–51. Springer, Heidelberg, April 2013.

[Alb17] Martin R. Albrecht. On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 103–129. Springer, Heidelberg, April / May 2017.

[ALM+92] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and hardness of approximation problems. In *33rd FOCS*, pages 14–23. IEEE Computer Society Press, October 1992.

[APS15] Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

[Bab85] László Babai. Trading group theory for randomness. In *17th ACM STOC*, pages 421–429. ACM Press, May 1985.

[Bac13] Adam Back. Bitcoins with homomorphic value (validatable but encrypted), October 2013. BitcoinTalk post, https://bitcointalk.org/index.php?topic=305791.0.

[Ban95] Wojciech Banaszczyk. Inequalities for convex bodies and polar reciprocal lattices inr n. *Discrete & Computational Geometry*, 13(2):217–231, 1995.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, Heidelberg, May 2005.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Heidelberg, August 2004.

[BBSU12] Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. Bitter to better - how to make Bitcoin a better currency. In Angelos D. Keromytis, editor, *FC 2012*, volume 7397 of *LNCS*, pages 399–414. Springer, Heidelberg, February / March 2012.

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

[BCC+14] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. The hunting of the SNARK. Cryptology ePrint Archive, Report 2014/580, 2014. http://eprint.iacr.org/2014/580.

[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 Shafi Goldwasser, editor, *ITCS 2012*, pages 326–349. ACM, January 2012.

[BCG+14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014.

[BCG+17] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, and Michele Orrù. Homomorphic secret sharing: Optimizations and applications. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 2105–2122. ACM Press, October / November 2017.

[BCI+10] Eric Brier, Jean-Sébastien Coron, Thomas Icart, David Madore, Hugues Randriam, and Mehdi Tibouchi. Efficient indifferentiable hashing into ordinary elliptic curves. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 237–254. Springer, Heidelberg, August 2010.

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 315–333. Springer, Heidelberg, March 2013.

[BCJ08] Ali Bagherzandi, Jung Hee Cheon, and Stanislaw Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008*, pages 449–458. ACM Press, October 2008.

[Bea96] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th ACM STOC*, pages 479–488. ACM Press, May 1996.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988.

[BFS16] Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 777–804. Springer, Heidelberg, December 2016.

[BG93] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 390–420. Springer, Heidelberg, August 1993.

[BG14] Shi Bai and Steven D. Galbraith. Lattice decoding attacks on binary LWE. In Willy Susilo and Yi Mu, editors, *ACISP 14*, volume 8544 of *LNCS*, pages 322–337. Springer, Heidelberg, July 2014.

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 509–539. Springer, Heidelberg, August 2016.

[BGI17] Elette Boyle, Niv Gilboa, and Yuval Ishai. Group-based secure computation: Optimizing rounds, communication, and computation. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 163–193. Springer, Heidelberg, April / May 2017.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 416–432. Springer, Heidelberg, May 2003.

[BHY09] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 1–35. Springer, Heidelberg, April 2009.

[BISW17] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Lattice-based SNARGs and their application to more efficient obfuscation. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 247–277. Springer, Heidelberg, April / May 2017.

[BISW18] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Quasi-optimal snargs via linear multi-prover interactive proofs. Cryptology ePrint Archive, Report 2018/133, 2018. https://eprint.iacr.org/2018/133.

[BLCL91] Gilles Brassard, Sophie Laplante, Claude Crépeau, and Christian Léger. Computationally convincing proofs of knowledge. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 251–262. Springer, 1991.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001.

[BLS03] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 257–267. Springer, Heidelberg, September 2003.

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 2004.

[BMP00] Victor Boyko, Philip D. MacKenzie, and Sarvar Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 156–171. Springer, Heidelberg, May 2000.

[BNM+14] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A. Kroll, and Edward W. Felten. Mixcoin: Anonymity for bitcoin with accountable mixes. In Nicolas Christin and Reihaneh Safavi-Naini, editors, *FC 2014*, volume 8437 of *LNCS*, pages 486–504. Springer, Heidelberg, March 2014.

[BNN07] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Unrestricted aggregate signatures. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *ICALP 2007*, volume 4596 of *LNCS*, pages 411–422. Springer, Heidelberg, July 2007.

[BOV03] Boaz Barak, Shien Jin Ong, and Salil P. Vadhan. Derandomization in cryptography. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 299–315. Springer, Heidelberg, August 2003.

[BP15] Nir Bitansky and Omer Paneth. ZAPs and non-interactive witness indistinguishability from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 401–427. Springer, Heidelberg, March 2015.

[BP17] Alessandro Budroni and Federico Pintore. Efficient hash maps to $\mathbb{G}_2$ on BLS curves. Cryptology ePrint Archive, Report 2017/419, 2017. http://eprint.iacr.org/2017/419.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737. Springer, Heidelberg, April 2012.

[BPW12] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the Fiat-Shamir heuristic and applications to Helios. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 626–643. Springer, Heidelberg, December 2012.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006.

[BSBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018. https://eprint.iacr.org/2018/046.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, October 2011.

[CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 3–33. Springer, Heidelberg, December 2016.

[Cha08] Melissa Chase. *Efficient non-interactive zero-knowledge proofs for privacy applications*. PhD thesis, Brown Univesity, 2008.

[Dam92] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 445–456. Springer, Heidelberg, August 1992.

[DDO+01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, Heidelberg, August 2001.

[DFGK14] George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 532–550. Springer, Heidelberg, December 2014.

[DGS+18] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: Bypassing internet challenges anonymously. *PoPETs*, 2018(3):164–180, July 2018.

[DM15] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 617–640. Springer, Heidelberg, April 2015.

[DN00] Cynthia Dwork and Moni Naor. Zaps and their applications. In *41st FOCS*, pages 283–293. IEEE Computer Society Press, November 2000.

[DS19] Alex Davidson and Nick Sullivan. The Privacy Pass Protocol. Internet-Draft draft-privacy-pass-00, Internet Engineering Task Force, November 2019. Work in Progress.

[FHSS+19] Armando Faz-Hernandez, Sam Scott, Nick Sullivan, Riad S. Wahby, and Christopher A. Wood. Hashing to Elliptic Curves. Internet-Draft draft-irtf-cfrg-hash-to-curve-05, Internet Engineering Task Force, November 2019. Work in Progress.

[FKR12] Laura Fuentes-Castañeda, Edward Knapp, and Francisco Rodríguez-Henríquez. Faster hashing to $\mathbb{G}_2$. In Ali Miri and Serge Vaudenay, editors, *SAC 2011*, volume 7118 of *LNCS*, pages 412–430. Springer, Heidelberg, August 2012.

[FMMO18] Prastudy Fauzi, Sarah Meiklejohn, Rebekah Mercer, and Claudio Orlandi. Quisquis: A new design for anonymous cryptocurrencies. Cryptology ePrint Archive, Report 2018/990, 2018. https://eprint.iacr.org/2018/990.

[FO18] Georg Fuchsbauer and Michele Orrù. Non-interactive zaps of knowledge. In Bart Preneel and Frederik Vercauteren, editors, *ACNS 18*, volume 10892 of *LNCS*, pages 44–62. Springer, Heidelberg, July 2018.

[FOS19] Georg Fuchsbauer, Michele Orrù, and Yannick Seurin. Aggregate cash systems: A cryptographic investigation of mimblewimble. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 657–689. Springer, Heidelberg, May 2019.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *22nd ACM STOC*, pages 416–426. ACM Press, May 1990.

[Fuc18] Georg Fuchsbauer. Subversion-zero-knowledge SNARKs. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 315–347. Springer, Heidelberg, March 2018.

[Gal13] Steven D. Galbraith. Space-efficient variants of cryptosystems based on learning with errors. preprint, 2013. https://www.math.auckland.ac.nz/~sgal018/compact-LWE.pdf.

[GCKG14] Arthur Gervais, Srdjan Capkun, Ghassan O. Karame, and Damian Gruber. On the privacy provisions of bloom filters in lightweight bitcoin clients. In Charles N. Payne Jr., Adam Hahn, Kevin R. B. Butler, and Micah Sherr, editors, *Annual Computer Security Applications Conference - ACSAC 2014*, pages 326–335. ACM, 2014.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013.

[GJS15] Qian Guo, Thomas Johansson, and Paul Stankovski. Coded-BKW: Solving LWE using lattice codes. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 23–42. Springer, Heidelberg, August 2015.

[GKM+18] Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Heidelberg, August 2018.

[GMNO18] Rosario Gennaro, Michele Minelli, Anca Nitulescu, and Michele Orrù. Lattice-based zk-SNARKs from square span programs. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 556–573. ACM Press, October 2018.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, December 1994.

[Gol93] Oded Goldreich. A uniform-complexity treatment of encryption and zero-knowledge. *Journal of Cryptology*, 6(1):21–53, March 1993.

[GOS06a] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 97–111. Springer, Heidelberg, August 2006.

[GOS06b] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 339–358. Springer, Heidelberg, May / June 2006.

[Gro06] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, Heidelberg, December 2006.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, December 2010.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.

[GS86] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *18th ACM STOC*, pages 59–68. ACM Press, May 1986.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008.

[Gt12] Torbjörn Granlund and the GMP development team. *GNU MP: The GNU Multiple Precision Arithmetic Library*, 5.0.5 edition, 2012. http://gmplib.org/.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011.

[HAB+17] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. TumbleBit: An untrusted bitcoin-compatible anonymous payment hub. In *NDSS 2017*. The Internet Society, February / March 2017.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[HJP13] W. Hart, F. Johansson, and S. Pancratz. FLINT: Fast Library for Number Theory, 2013. Version 2.4.0, http://flintlib.org.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 145–161. Springer, Heidelberg, August 2003.

[Jed16] Tom Elvis Jedusor. Mimblewimble, 2016. Available at https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.txt.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992.

[KK13] Vladimir Kolesnikov and Ranjit Kumaresan. Improved OT extension for transferring short secrets. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 54–70. Springer, Heidelberg, August 2013.

[KKM14] Philip Koshy, Diana Koshy, and Patrick McDaniel. An analysis of anonymity in bitcoin using P2P network traffic. In Nicolas Christin and Reihaneh Safavi-Naini, editors, *FC 2014*, volume 8437 of *LNCS*, pages 469–485. Springer, Heidelberg, March 2014.

[KM07] Neal Koblitz and Alfred J. Menezes. Another look at "provable security". *Journal of Cryptology*, 20(1):3–37, January 2007.

[KN08] Gillat Kol and Moni Naor. Cryptography and game theory: Designing protocols for exchanging information. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 320–339. Springer, Heidelberg, March 2008.

[KW93] M. Karchmer and A. Wigderson. On span programs. In IEEE Computer Society Press, editor, *In Proc. of the 8th IEEE Structure in Complexity Theory*, pages 102–111, Gaithersburg, MD, USA, 1993. IEEE Computer Society Press.

[LMRS04] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. Sequential aggregate signatures from trapdoor permutations. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 74–90. Springer, Heidelberg, May 2004.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 319–339. Springer, Heidelberg, February 2011.

[Max13a] Gregory Maxwell. CoinJoin: Bitcoin privacy for the real world, August 2013. BitcoinTalk post, https://bitcointalk.org/index.php?topic=279249.0.

[Max13b] Gregory Maxwell. Transaction cut-through, August 2013. BitcoinTalk post, https://bitcointalk.org/index.php?topic=281848.0.

[Max15] Gregory Maxwell. Confidential Transactions, 2015. Available at https://people.xiph.org/~greg/confidential_values.txt.

[MGGR13] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed E-cash from Bitcoin. In *2013 IEEE Symposium on Security and Privacy*, pages 397–411. IEEE Computer Society Press, May 2013.

[Mic94] Silvio Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, November 1994.

[MPJ+13] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. A fistful of bitcoins: characterizing payments among men with no names. In Konstantina Papagiannaki, P. Krishna Gummadi, and Craig Partridge, editors, *Internet Measurement Conference, IMC 2013*, pages 127–140. ACM, 2013.

[Nak08] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, 2008. Available at http://bitcoin.org/bitcoin.pdf.

[Nao03] Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 96–109. Springer, Heidelberg, August 2003.

[OOS17] Michele Orrù, Emmanuela Orsini, and Peter Scholl. Actively secure 1-out-of-N OT extension with application to private set intersection. In Helena Handschuh, editor, *CT-RSA 2017*, volume 10159 of *LNCS*, pages 381–396. Springer, Heidelberg, February 2017.

[Ped92] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013.

[Poe16] Andrew Poelstra. Mimblewimble, 2016. Available at https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.pdf.

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

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, Heidelberg, August 2008.

[Ràf15] Carla Ràfols. Stretching groth-sahai: NIZK proofs of partial satisfiability. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 247–276. Springer, Heidelberg, March 2015.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[RMK14] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. CoinShuffle: Practical decentralized coin mixing for bitcoin. In Miroslaw Kutylowski and Jaideep Vaidya, editors, *ESORICS 2014, Part II*, volume 8713 of *LNCS*, pages 345–364. Springer, Heidelberg, September 2014.

[Rog15] Phillip Rogaway. The moral character of cryptographic work. Cryptology ePrint Archive, Report 2015/1162, 2015. http://eprint.iacr.org/2015/1162.

[RS13] Dorit Ron and Adi Shamir. Quantitative analysis of the full Bitcoin transaction graph. In Ahmad-Reza Sadeghi, editor, *FC 2013*, volume 7859 of *LNCS*, pages 6–24. Springer, Heidelberg, April 2013.

[RTRS18] Tim Ruffing, Sri Aravinda Thyagarajan, Viktoria Ronge, and Dominique Schröder. Burning Zerocoins for Fun and for Profit: A Cryptographic Denial-of-Spending Attack on the Zerocoin Protocol. IACR Cryptology ePrint Archive, Report 2018/612, 2018. Available at https://eprint.iacr.org/2018/612.

[SBC+09] Michael Scott, Naomi Benger, Manuel Charlemagne, Luis J. Dominguez Perez, and Ezekiel J. Kachisa. Fast hashing to G_2 on pairing-friendly curves. In Hovav Shacham and Brent Waters, editors, *PAIRING 2009*, volume 5671 of *LNCS*, pages 102–113. Springer, Heidelberg, August 2009.

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, Heidelberg, August 1990.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991.

[Sha90] Adi Shamir. IP=PSPACE. In *31st FOCS*, pages 11–15. IEEE Computer Society Press, October 1990.

[Sho01] Victor Shoup. OAEP reconsidered. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 239–259. Springer, Heidelberg, August 2001.

[SMD14a] Amitabh Saxena, Janardan Misra, and Aritra Dhar. Increasing anonymity in bitcoin. In Rainer Böhme, Michael Brenner, Tyler Moore, and Matthew Smith, editors, *FC 2014 Workshops*, volume 8438 of *LNCS*, pages 122–139. Springer, Heidelberg, March 2014.

[SMD14b] Amitabh Saxena, Janardan Misra, and Aritra Dhar. Increasing Anonymity in Bitcoin. In Rainer Böhme, Michael Brenner, Tyler Moore, and Matthew Smith, editors, *1st Workshop on Bitcoin Research - Bitcoin 2014*, volume 8438 of *LNCS*, pages 122–139. Springer, 2014.

[Ste03] Jacques Stern. Why provable security matters? (invited talk). In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 449–461. Springer, Heidelberg, May 2003.

[SZ16] Yonatan Sompolinsky and Aviv Zohar. Bitcoin's Security Model Revisited, 2016. Manuscript available at http://arxiv.org/abs/1605.09193.

[vS13] Nicolas van Saberhagen. CryptoNote v 2.0, 2013. Manuscript available at https://cryptonote.org/whitepaper.pdf.

[Wee07] Hoeteck Wee. Lower bounds for non-interactive zero-knowledge. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 103–117. Springer, Heidelberg, February 2007.

RÉSUMÉ

Cette thèse étudie les arguments de connaissance non-interactives, une brique de base utilisé en cryptographie quipermet à un *fuornisseur de preuve* de convaincre un *vérificateur* qu'une proposition est vraie sans révéler d'autres informations que la véracité de la proposition. Notre attention se porte en particulier sur la constructions des nouveaux arguments avec des garanties de sécurité plus fortes: d'abord, nous montrons l'existence des arguments de connaissance non-interactives et *witness-indistinguishable* dans le modèle standard (sans oracle aléatoire ni chaîne de référence commune). Ensuite, nous étudions la sécurité post-quantique des SNARKs, une famille de schémas de preuve de connaissance succincte. Enfin, nous analysons la sécurité d'une crypto monnaie anonyme dont la sécurité est garantie par des arguments de connaissance: Mimblewimble. La cryptomonnaie était proposée par un auteur anonyme en 2016, et nous fournissons la première analyse formelle de sa sécurité.

MOTS CLÉS

cryptographie, peuves de connaissance, preuves à divulgation nulle, post-quantique, cryptomonnaie.

ABSTRACT

This thesis studies non-interactive arguments of knowledge, a cryptographic primitive that allows a prover to convince a verifier of the truth of a certain statement. It focuses on cryptographic constructions that allow a user to prove knowledge of a so-called witness x that satisfies a circuit $\mathtt{C}$, while simultaneously hiding it.

First, we prove the existence of non-interactive witness-indistinguishable arguments of knowledge in the standard model. Our proof system is an argument of knowledge that is secure even if an adversary subverts the initial parameters. Secondly, we revisit a family of zero-knowledge arguments of knowledge (SNARKs), and show that it can be moved to post-quantum assumptions, as long as the verifier is known in advance. Lastly, we consider a novel, anonymous cryptocurrency whose security can be guaranteed via arguments of knowledge: Mimblewimble. The cryptocurrency was proposed by an anonymous author in 2016. We provide the first formal analysis of it, fixing a security issue present in the initial proposal.

KEYWORDS

cryptography, proof of knowledge, zero-knowledge proof, cryptocurrency, post-quantum.