Cryptography for Scalability and Identity in Blockchain Applications

Hamidreza Khoshakhlagh

PhD Dissertation

Department of Computer Science
Aarhus University
Denmark

Cryptography for Scalability and Identity in Blockchain Applications

A Dissertation
Presented to the Faculty of Natural Sciences
of Aarhus University
in Partial Fulfillment of the Requirements
for the PhD Degree

by
Hamidreza Khoshakhlagh
May 31, 2022

Abstract

Blockchains are distributed systems of increasing importance. They work by recording transactions between mutually distrusting parties in an immutable manner and without requiring a trusted third party. Their benefits, however, come at the expense of privacy and scalability. The history of all transactions should be exposed to the network making it possible to violate the privacy and anonymity of parties. Furthermore, verifying transactions requires an excessive amount of computation, because verification time is not independent of the computation. This creates a scalability issue.

Several approaches have been proposed so far to address these two challenges. To ensure privacy, cryptocurrencies like Zerocash provide fully privacy-preserving payments; i.e., they hide both the transaction value and the identities of sender and receiver. While being a satisfactory solution for the privacy challenge, such approaches are not compliant with current regulations which require some form of accountability to prevent misuse of the system. To ensure scalability, zero-knowledge succinct non-interactive argument systems, or zkSNARKs have been used by many decentralized systems in the last decade. While being an attractive tool for not only scalability but also privacy, the security definitions of commonly-used zkSNARKs are either based on unrealistic trust assumptions, or are formalized without taking real-world scenarios into account. In this thesis, we show how to overcome the aforementioned shortcomings.

Our first contribution comprises designing a blockchain identity management system that provides privacy and accountability in a balanced manner. We develop cryptographic mechanisms that enhance accountability against misuse of the blockchain, while still ensuring privacy.

The second contribution is the study of zkSNARKs with desirable features that significantly reduce the trust assumptions in the setup phase and ensure security in the strongest (but also most realistic) setting. Specifically, we show zkSNARKs that enjoy universality and updatability of the structured reference string (SRS), and further simulation extractability out-of-the-box.

Seeing anonymity as a tool of privacy in blockchain systems, the third result is related to the task of transferring secrets in the so-called "you only speak once" (YOSO) model, in which the adversary has strong denial of service (DoS) capabilities and can identify and block parties as soon as their identity become known by sending a message.

Our fourth contribution is in the domain of predictable arguments. Predictable arguments are private-coin argument systems where the answer of the prover can be predicted by the verifier. In this part, we study predictable arguments with additional privacy properties such as zero-knowledge and witness indistinguishability.

Resumé

Blockchains er distribuerede systemer af stadig større betydning. De fungerer ved at bogføre transaktioner blandt parter uden gensidig tillid til hinanden i en "uforanderlig tinglysningsbog" og opnår dette uden at afhænge af en betroet tredjepart. Deres fordele kommer dog på bekostning af "privacy" og skalerbarhed: den samlede historie over alle foretaget transaktioner afsløres overfor netværket, hvilket fører til dårlig anonymitet og "privacy" for brugerne. Dertil kommer at verifikationen af samtlige transaktioner kræver betydelig regnekraft fordi verifikations tiden ikke er uafhængig af den underliggende beregnings længe, hvilket forrudsager problemer med systemets skalerbarhed.

Et antal teknikker er i tidligere litteratur blevet foreslået for at imødegå disse to udfordringer. For at forbedre "privacy" ved at skjule transaktionernes indhold tilbyder kryptovaluta såsom Zerocash "fully privacy-preserving payments", hvor både beløbet, samt betalerens og modtagerens identitet skjules fuldstændigt. Mens dette fører til en tilfredsstillende løsning ud fra et privatlivsmæssigt synspunkt, er sådanne teknikker ikke forenelige med nuværende lovgivning, som fordre at systemet implementere beskyttelse mod misbrug (eksempelvis "Know Your Customer"). For at adressere problemerne med skalerbarhed har "zero-knowledge Succinct Non-interactive ARguments of Knowledge" eller "zkSNARKS" været benyttet af mange decentraliserede systemer i løbet af det seneste årti. Tilltrods for at dette værktøj tillader løsniger på både "privacy" og skalerbarheds problematikkerne, er sikkerhedsdefinitionerne af ofte benyttede zkSNARKS enten baseret på urealistiske tillidsantagelser eller formaliseret uden at tage højde for real-world scenarier. I denne afhandling, vil vi redegøre for hvorledes disse begrænsninger kan overkommes.

Vores første bidrag består i et design af et "blockchain identity management" system der imødekommer både retten til privatliv og krævet om juridisk ansvar, på en balanceret måde. Vi udvikler kryptografiske mekanismer der bedre forhindre misbrug, men samtidig beskytter brugernes privatliv.

Det andet bidrag består i et studie af zkSNARKS med ønskværdige egenskaber der reducere tillidsantagelserne betydelig og sikre sikkerheden i den stærkeste (men også realistiske) scenario. Mere specifikt, beskriver vi zkSNARKS der har en universel og opdaterbar "Structured Reference String" (SRS), samt tillader "simulation extractability".

Ud fra betragtningen af anonymitet som et "privacy tool" i blockchain systemer, omhandler det tredje resultat overførselen af af hemmeligheder i den såkaldte “You Only Speak Once” (YOSO) model, hvor angriberen har potente "Denial of Service" (DoS) evner som tillader ham at blokere parter så snart at deres indentitet bliver kendt efter at havde sendt blot én enkelt besked.

Vores fjerde bidrag er indenfor "predicatable arguments". "Predictable arguments" er "private-coin" argumenter hvor beviserens svar kan forudsiges af verifieren. I dette afsnit, betragter vi predicatable arguments med yderligere privacy egenskaber så som "zero-knowledge" og "witness indistinguishability".

Acknowledgments

I am deeply thankful to my advisor Jesper Buus Nielsen for his amazing job as an advisor, and for his patience and confidence in me. I am especially grateful to him for giving me great freedom to discover what I enjoy.

My greatest respect and gratitude to Matteo Campanelli and Chaya Ganesh who played an important role during my PhD research. Chaya was the one to whom I always felt comfortable asking my "stupid" questions. She was always supportive and patient, and had time to help and offer advice. Matteo stood next to me during the difficult times of my PhD years. I was greatly helped by him to practice skills a PhD student should have, and his critical and insightful comments provided me the opportunity to grow as a student. I will be grateful to both of them forever.

I would like to thank all my amazing co-authors: Ivan Damgård, Chaya Ganesh, Claudio Orlandi, Luisa Siniscalchi, Matteo Campanelli, Bernardo David, Anders Konring, Jesper Buus Nielsen, Markulf Kohlweiss, Anca Nitulescu, Michal Zajac, Behzad Abdolmaleki, Helger Lipmaa, Daniel Slamanig, Janno Siim, and Roberto Parisella. I have learned a lot from all of you and none of my research would have been possible without your encouragement and assistance.

I am very grateful to Dario Fiore who gave me an opportunity to visit their research group at IMDEA in Madrid and collaborate with them.

Many thanks to Mathias Hall-Andersen for taking the time to translate my abstract into Danish. I would also like to thank all the people from the Aarhus Crypto Group for creating an encouraging environment, for many enjoyable conversations, and for all their support during difficult times.

I dedicate this dissertation to my parents who have supported me all the way since the beginning of my studies. I also dedicate this dissertation to my wife for all of her love, support and constant encouragement. I do not think I could have done my PhD in a foreign country without counting on her.

Hamidreza Khoshakhlagh,
Aarhus, May 31, 2022.

Contents

Part I

Overview

Chapter 1

Introduction

Thanks to Bitcoin and other cryptocurrencies, distributed ledgers and blockchain has gotten a lot of attention in recent years. To put it in the simplest terms, a distributed ledger can be seen as a mechanism that maintains data across a distributed database while ensuring the same view of data for all honest parties, even in the presence of corrupt parties. Blockchain is most simply defined as a digitally distributed ledger that records transactions between different parties in a verifiable manner. It eliminates the need for verification by a central authority (e.g., trusted third party) by allowing users to maintain a copy of the ledger, and then synchronizing all copies by means of a consensus algorithm. While digital currency is the most popular, and also the first implementation of blockchain, the technology can provide many other applications. It enables smart contracts, decentralized government services, and all types of transactions in a more efficient and robust manner than the traditional systems that their centralized nature enables third-parties to collect and control massive amounts of personal data, causing a privacy breach.

Despite the aforementioned benefits, there are several limitations in blockchain technology that need to be taken into account, the most important ones being *privacy* and *scalability*.

Privacy issue

In blockchain, the history of all events should be available for everyone to read and this limits the applications of this technology. For example, in the first generation of cryptocurrencies such as Bitcoin and Ethereum, such data availability makes it possible to monitor particular payment addresses for a long time and then trace any movement, hence violating the privacy and anonymity of the target user. To ensure privacy and anonymity for their users, new system with privacy guarantees are proposed [25, 82, 144]. Cryptocurrencies like Zerocash [25] provide fully privacy-preserving payments; i.e., they hide both the transaction value and the identities of sender and receiver. Smart contract systems like Hawk [128] enable privacy-preserving state transitions, where the application's data is hidden from third parties.

Privacy versus Accountability. Complete privacy and anonymity perfectly reduces the traces by the system users and therefore is a satisfactory solution for the privacy issue. However, it hinders the most blockchain use-cases in legislation settings that need some form of *accountability*. For example, to identify ill-intentioned individuals who misuse the system by participating in illegal activities such as trading of illegal substances, tax-evasion, money laundering, etc, financial institutions must conform with regulations like "Know your customer" (KYC) and

"Anti-money laundering" (AML) that allow a *legal authority* to learn transaction details. The lack of this property in existing privacy-preserving systems in fact explains why traditional financial institutions have so far not been convinced about the blockchain technology's potential in financial services.

The seemingly contradictory requirements of user anonymity, and the aforementioned regulations at the same time seems like a major hurdle in widespread adoption of the blockchain. Most existing systems either provide full privacy and anonymity without accountability in mind (like ZCash [25]), or they are based on approaches that require a single trusted party (i.e., bank) like [57]. Having a form of accountability and *identity management* not only ensures anonymity of honest users, but also opens up the possibility of identifying malicious users who look at cryptocurrencies as a way of facilitating illegal activities.

Anonymity. While ensuring user privacy in the blockchain setting seems to be straightforward using known cryptographic techniques, achieving full-anonymity as required in some applications is challenging. Imagine a blockchain setting with thousands of users that have publicly known identities and want to perform some computation in the presence of a powerful denial of service (DoS) adversary. In this setting, the adversary by knowing parties' identities can block parties as soon as they execute the protocol. Hence, to carry out the computation securely in the presence of such adversaries, one requires a new model of communication channels between users in which parties remain anonymous up until they carry out a task (i.e., *speak*), and as soon as they speak again, the adversary can identify and block them.

To capture this setting, the authors in [102] proposed a model, the so-called *you only speak once* (YOSO) model, in which the adversary with strong denial of service (DoS) capabilities monitors a communication network between a large number of *one-time* stateless parties—so-called *roles* in [102]—and mounts a DoS attack on a (role-assigned) *machine* as soon as it sends a message.

An interesting technical problem to deal with in the YOSO model is to find a way to map random unknown machines M to roles R in the protocol. A role can be "party 7 in round 42 of protocol X" being executed by some yet unknown machine. If it was always known that one particular machine executes a given role it makes it easy to mount a denial of service attack just before the role is to be executed. Second, if some future role in the protocol wants to send a secret message to the machine M which executes a future role R, then we somehow need a public key of M to become known without the identity of M becoming known. The aspect of how to make it possible to send a message to a future role without anyone knowing which physical machine is going to execute the role is called the role assignment (RA) aspect.

Existing solutions for role assignment are all unsatisfactory. One trivial solution is to use a generic witness encryption scheme [98] for an NP relation that defines the role assignment. However, existing witness encryption schemes are either based on strong assumptions such as multilinear maps [98, 100], or rely on heavy tools like indistinguishability obfuscation (iO) [97] which are currently not well understood.

A simpler (interactive) solution can be based on the approach proposed in [32, 101, 112] over a public blockchain where a nominated set of parties called *committee members* keep a message alive by resharing it from time to time up to the future point where the message is finally handed out to the receiver. As a more efficient solution, one can also sample a fresh key pair $(\mathsf{sk}_\mathsf{R}, \mathsf{pk}_\mathsf{R})$ for a PKE scheme, broadcast $(\mathsf{R}, \mathsf{pk}_\mathsf{R})$ on the blockchain and send the secret key sk_R to a *random* M without leaking M's identity. The public key pk_R creates a target-anonymous

channel to M and allows parties to send any message m to M by posting encryption of m (under pk_{R}) on the blockchain. The issue with this approach is that it requires a committee who takes care of key generation and selecting M. Further, they can only be used to select M according to an *already known* probability distribution. This signifies that such approaches are not applicable when the party selection for a given role is w.r.t. a future stake distribution.

Scalability issue

In public blockchain networks where all data are publicly accessible, transactions must be validated and processed at every node in the network. In other words, to maintain a public blockchain, each of the thousands of users must first receive and validate every transaction, and then permanently store them. This raises serious issues in scalability as e.g., clients who want to catch up on the latest state of the chain should spend significant amount of resources in terms of computation and bandwidth.

To address scalability issue in blockchain, Succinct Non-interactive ARguments of Knowledge (SNARKs) have been adopted by many real-world applications such as decentralized systems in the last decade. Zero-knowledge SNARKs (zkSNARKs) are SNARKs that further ensure privacy by satisfying a zero-knowledge property. In more details, a zkSNARK is a proof system that allows a prover to generate short and efficiently verifiable proofs without revealing anything more than the correctness of a statement. A series of works focused on constructing and implementing zkSNARKs for blockchain use-cases, such as cryptocurrencies. Zerocash [25] showed how to use zkSNARKs in distributed ledgers to achieve privacy-preserving digital payment systems.

Some examples of practical use of zkSNARKs are in the context of Zcash or in Ethereum system[1] for boosting the privacy and scalability of smart contracts. A different important SNARK application in blockchain is the Filecoin System[2] that implements a decentralized storage solution by means of Proof of Space via SNARKs. It is important to note that in all these usages, proofs are posted on the public chain so that any user can check that a statement is true (e.g., validity of a transaction, claims of storage etc.), while ideally expending few resources.

Despite their massive adoption in practice, zkSNARK schemes used today are still not well adapted or defined accordingly for the needs of such applications. More concretely, the security definitions are based on unrealistic trust requirements or are formalized without taking into consideration real-world scenarios. Most SNARK schemes require the generation of trusted setups which are hard to consider in practice. Another concern is that classical soundness definitions for SNARKs are not adapted to the decentralized scenarios, so there is still the need to handle additional security requirements, e.g. non-malleability of proofs is achieved by composing SNARKs with additional signature schemes. This introduces another concern: composing SNARKs with other cryptographic primitives had been shown not always to be secure [36, 85]. When using SNARKs in larger cryptographic protocols in real-world scenarios, adversarial provers may get additional information which can contribute to the generation of cheating proofs. To address this problem, stronger, and more useful, definitions of proof of knowledge were defined in the literature, but they also introduce other subtleties.

[1] Zcash `https://z.cash/`, Ethereum `https://ethereum.org`

[2] Filecoin, `https://filecoin.io`

Simulation extractability. Most zkSNARKs are shown to satisfy a standard knowledge soundness property. However, deployments of zkSNARKs in real-world applications require a stronger property – *simulation-extractability* (SE). This is because, in practice, an adversary against the zkSNARK has access to proofs provided by other parties using the same zkSNARK. For instance, in applications of zkSNARKs in privacy-preserving blockchains, proofs are posted on the chain for all blockchain-participants to see. Therefore, it is necessary for a zero-knowledge proof system to be resilient against adversaries that additionally get to see proofs generated by different parties before trying to forge.

Related Work. There exist many results on non-interactive zero-knowledge proofs (NIZKs) with simulation extractability. Groth [114] observed that a (black-box) SE NIZK is universally-composable (UC) [51]. In [78], Dodis et al. introduced a notion of (black-box) *true simulation extractability* and showed that this property is necessary for NIZKs to be UC-secure. In the context of zkSNARKs, Groth and Maller [117] proposed the first SE zkSNARK. Kosba's et al. [127] gave a general transformation from a NIZK to a black-box SE NIZK. Their transformation works also for zkSNARKs, although succinctness is not preserved anymore. In another transformation recently proposed by Abdolmaleki et al. [5], the authors show how to obtain non-black-box simulation extractability that preserves succinctness of the argument as well. By introducing minor modifications to the zkSNARK construction of [116] and making stronger assumptions, the authors of [39] showed that the resulting zkSNARK is SE. Recently, the work of [16] showed that the original Groth's proof system in [116] is weakly SE.

The challenge in the updatable SRS setting. One of the downsides of efficient zkSNARKs such as [72, 99, 115, 116, 129, 130, 141] is that they require a *trusted setup*, wherein a structured reference string (SRS) should be generated by a trusted party. This assumption, however, is not well founded in practice; if the party who generates the SRS is dishonest, they can produce proofs of invalid statements. That is, the knowledge soundness breaks down if the trusted setup assumption does not hold. To address this challenge, Groth et al [120] proposed a setting which allows parties – provers and verifiers – to take a current SRS and *update* it to a new SRS by contributing to its randomness in a verifiable way. The guarantee in this *updatable setting* is that knowledge soundness holds as long as one of the parties who updates the SRS is honest. The SRS is also *universal*, in that it does not depend on the relation to be proved but only on an upper bound in the size of the statements. While the SNARK of [120] is inefficient, as the SRS length is quadratic in the size of the statement, soon after, Maller et al. in [132] proposed Sonic the first universal zkSNARK with updatable and linear size SRS. Subsequently, Gabizon et al. designed Plonk [91] which currently is the most efficient updatable universal zkSNARK. Independently, Chiesa et al. [62] proposed Marlin with comparable efficiency to Plonk.

The notion of simulation-extractability for zkSNARKs which is well motivated in practice has not been studied in this updatable setting. As it turns out, defining SE for updatable SRS zkSNARKs requires some care. Since the SRS is being continually updated, it is possible for the adversary to see proofs with respect to *different* SRS'es before attempting to forge a proof with respect to a current SRS. A definition of SE in the updatable setting should hence take into account this additional power of the adversary and clarify with respect to which SRS (or SRS'es) the simulated proofs are generated and given to the adversary. This is not captured by existing definitions of SE.

Chapter 2

Our Contributions

This chapter outlines the main results of this thesis. The thesis is based on four papers [49, 71, 93, 124] that were written during my doctorate.

Our contributions can be summarized as follows:

- Balancing Privacy and Accountability in Blockchain [71]. This work proposes a novel design for identity management in Blockchain systems. The goal of the design is to develop cryptographic mechanisms that enhance accountability against misuse of the blockchain, while still ensuring privacy. My contribution to this paper was the security analysis of the construction in the Universal Composability (UC) framework.

- What Makes Fiat–Shamir zkSNARKs (Updatable SRS) Simulation Extractable? [93]. In this work, we study the notion of (updatable) simulation extractability for multi-message protocols that are in the random oracle model (ROM) and use a structured reference string (SRS) in the updatable setting. We show sufficient conditions for compiling such interactive protocols into simulation-extractable NIZK proof systems via the Fiat–Shamir transformation.

- Encryption to the Future [49]. This work investigates the problem of role assignment in the YOSO model [102]. In this work, we initiate the study of Encryption to the Future (EtF) as a cryptographic primitive and study how to encrypt to a future role with a strong security guarantee.

- (Commit-and-Prove) Predictable Arguments with Privacy [124]. This work studies predictable arguments with privacy properties such as zero-knowledge and witness indistinguishability. It also proposes a relaxed notion of predictability called *commit-and-prove* predictable arguments (CPPA), where all but the first message of the prover can be predicted. By the fact that predictable arguments and witness encryption schemes are equivalent notions, CPPA can be seen as a variant of witness encryption we introduce in [49] that provides decryptor's privacy as well.

In the rest of this section, we dive a bit more in-depth into the details of these works. The chapter concludes with a short overview of other research projects I have done during my PhD study and a short section introducing general notation used in this thesis.

2.1 Chapter 3: Balancing Privacy and Accountability in Blockchain Identity Management

In this section, we summarize our results on balancing privacy and accountability in blockchain identity management presented in [70, 71].

The main goal of this work is to answer the question of how to making a balance between privacy and accountability in blockchain systems. To this end, the work proposes a new design of an *identity layer* that provides privacy for the users—that is, no one can learn the identity of the account owners by observing the network transactions. Further, it achieves accountability; i.e., in the presence of reasonable suspicion, authorized parties can revoke the anonymity of the account owners and access to the history of their transactions.

The proposed identity management system involves three players: 1. *Account Holders* (AH) as network participants who are interested in creating accounts on the blockchain and performing transactions. 2. *Identity Providers* (IP) as authorized parities who verify and store the identity and credentials of account holders. And 3. *Anonymity Revokers* (AR) as another type of authorized parities who can—if more than a threshold—revoke the anonymity of suspicious account holders. To limit the number of accounts an AH can open, the system defines a parameter $\mathsf{Max}_{\mathsf{ACC}}$ as the maximum number of possible accounts for every AH and forces the AH to use an account identifier $\mathsf{RegID}_{\mathsf{ACC}_i}$ for its i-th account that is derived from a PRF applied to integer $1 \leq i \leq \mathsf{Max}_{\mathsf{ACC}}$. To this end, the AH, in an initial registration with the IP, selects a PRF key K and sends a threshold encryption of K (under AR's public keys) to the IP. This ciphertext can later be decrypted by a sufficient number of ARs and obviously allows them to learn all the accounts belonging to a suspicious AH. After the initial registration with an IP, account holders can non-interactively create accounts. An account includes some data from the AH such as a threshold encryption of the AH's public key (also stored with the IP at registration time) and zero-knowledge proofs that attest the attributes the AH has chosen to publish in the account have been signed by an IP, and that the account identifier has been computed correctly. An account also includes various account-specific public keys that allow the AH to perform anonymous transactions in the network. In case of suspicion to an account (e.g., account being used for fraudulent purposes), the encrypted account information can be decrypted by a qualified set of the ARs who can then link the account to an AH registered with an IP.

In our construction, we use Pedersen commitments [143] for the underlying commitment scheme and Dodis-Yampolskiy PRF [77] to generate account identifiers. While this PRF is secure only for small domains, it is sufficient for our design as the maximum number of accounts an AH can open is upper-bounded by a constant $\mathsf{Max}_{\mathsf{ACC}}$. We use Pointcheval-Sanders (PS) blind signature scheme [145] which allows an IP to issue credentials to AHs. To encrypt field-element plaintexts so that proving statements about the message is easy using Σ-protocols, we use CL encryption scheme [54]. Despite having Elgamal-like structure, this scheme is different from Elgamal encryption, where only limited-size plaintexts can be decrypted efficiently. This useful property will also be used in our construction of black-box extractable (succinct) NIZK proofs in the CRS model, wherein the prover CL encrypts the witness under a public key that is part of the CRS. In particular, while our generic lifting transform of Fiat-Shamir NIZKs for DL-languages into UC NIZKs is a folklore technique [75], using CL encryption scheme is novel to the best of our knowledge. Finally, we use techniques related to zkSNARKs on committed/signed messages in the spirit of [9, 48]. Such techniques allow us to use SNARKs only on small circuits, thus achieving much better efficiency compared to naive approach of converting our large statements

(e.g, group operations) into a Boolean circuit to be evaluated inside the SNARK.

We prove the security of our constructions in the Universal Composability (UC) framework, where either any number of account holders are actively corrupt, or the identity providers are semi-honest corrupt. Depending on the particular configuration, the system can tolerate different corruption levels among the anonymity revokers. That is, it provides security in the presence of actively corrupt users and a threshold number of passively corrupt anonymity revokers; or, in the presence of passively corrupt identity-providers and a threshold number of passively corrupt anonymity revokers.

Bibliography information

Ivan Damgård, Chaya Ganesh, Hamidreza Khoshakhlagh, Claudio Orlandi, and Luisa Siniscalchi. Balancing privacy and accountability in blockchain identity management. pages 552–576, 2021. doi: 10.1007/978-3-030-75539-3_23

2.2 Chapter 4: What Makes Fiat–Shamir zkSNARKs (Updatable SRS) Simulation Extractable?

This section summarizes our results on simulation extractability of Fiat-Shamir zkSNARKs in the updatable setting, presented in [93]. In a nutshell, there exist several approaches to make current zkSNARKs adequate for real-life deployment among which are *simulation extractability* (SE) and *updatability*. There are zkSNARKs that are SE and zkSNARKs that are updatable universal. In this work, we study zkSNARKs that enjoy both of these properties. A more detailed description follows.

Motivated by applications such as privacy-preserving cryptocurrencies [25], and uses in the context of Zcash, Ethereum system[1] for privacy and scalability of smart contracts that demand short proofs and fast verification, there has been a series of works on constructing zkSNARKs.

While the two aforementioned properties make zkSNARKs useful in real-life system deployment, e.g. [15, 64, 65, 148, 158], an important question concerns if existing security models for zkSNARKs are good enough for real world applications. This becomes even more important when one realizes that zkSNARKs with constant-sized proofs have been constructed in the common reference string (CRS) model where the CRS used by the parties—i.e. provers and verifiers—comes with a trapdoor which can be used to break soundness; i.e., a malicious prover can convince a verifier to accept a false statement. It is therefore very important to make sure that the CRS trapdoor is not in the possession of a particular party, or has not been leaked by the CRS generator. In the context of distributed systems e.g., blockchain, this becomes even more challenging where assuming a trusted party for CRS generation conflicts with the whole purpose of decentralized systems.

The seminal work of Bellare et al. [22] tackled this question and proposed notions of *subversion zero knowledge* and *subversion soundness* which informally state that these properties hold even when the CRS is generated maliciously. Importantly, they show that that no system can be simultaneously subversion zero-knowledge and subversion-sound. Abdolmaleki et al. [4],

[1] Zcash `https://z.cash/`, Ethereum `https://ethereum.org`

and independently Fuchsbauer [88] later showed that the most efficient zkSNARK for QAP[2] by Groth [116] can be made subversion zero-knowledge by making minor modifications to its CRS and without sacrificing its efficiency.

Although efficient, Groth's zkSNARK comes with a drawback—the CRS depends on the relation. That is, to show the correct evaluations of two different arithmetic circuits, one has to generate the proofs using two different CRS-s, one for each circuit. The aforementioned troublesomeness of CRS generation in distributed setting makes it very desirable to have a type of CRS that is circuit-independent. That is, a single CRS is *universal* and can be utilized for all circuits of a given size. One of the first such universal zkSNARKs was proposed by Groth et al. [120] where the proof system, in addition to CRS universality, also ensures a strong security property called *updatable soundness*. The notion of updatable soundness captures a setting where zkSNARK provers and verifiers can *update* the CRS so that as long as there is at least one honest CRS updating, the proof system guarantees (knowledge) soundness. While the construction of [120] comes with a quadratic-size CRS and thus inefficient, updatable universal zkSNARKs with short CRS such as Sonic proposed by Maller et al. in [132], Plonk proposed by Gabizon et al. [91], and Marlin proposed by Chiesa et al. [62] were proposed soon enough after this work. Interestingly, all these works follow a rather modular blueprint that consists of two parts: first, designing an information theoretic interactive protocol, or more precisely, an algebraic variant of an Interactive Oracle Proof (IOP). Second, compiling the IOP via a cryptographic tool (i.e., polynomial commitment (PC) schemes) to obtain an interactive argument system. This is then later turned into a SNARK using the Fiat-Shamir transformation in the Random Oracle Model (ROM).

On the importance of simulation extractability. Although zkSNARKs are shown to satisfy a (standard) knowledge soundness definition, simulation-extractability (SE) is the property that should be required from zkSNARKs used in practice. This is since, arguably, in the real life one simply cannot assume that the adversary who tries to break security of a system does not have access to any proofs provided by other parties using the same zero-knowledge scheme. On the contrary, in the most popular applications of zkSNARKs, like privacy-preserving blockchains, proofs made by all blockchain-participants are usually public. Thus, it is only reasonable to require a zero-knowledge proof system to be resilient to attacks that utilise proofs generated by different parties.

There are many results on simulation extractable non-interactive zero-knowledge proofs (NIZKs). Groth [114] first showed that black-box simulation extractable NIZK is universally-composable (UC) [51]. In the context of zkSNARKs, Groth and Maller [117] proposed the first simulation-extractable zkSNARK. Later, Kosba's et al. [127] gave a general transformation from a NIZK to a black-box SE NIZK. Applying their transformation to zkSNARKs however kills succinctness of the proof system. In a recent work, Abdolmaleki et al. [5] showed another transformation using similar techniques that obtains non-black-box simulation extractability but also preserves succinctness. Independently, some works has focused on making known zkSNARKs simulation extractable by adding minor modifications [11, 39].

State of the art—simulation-extractable updatable universal zkSNARKs. There are zkSNARKs that are simulation-extractable and zkSNARKs that are universal, however there are

[2]QAP stands for Quadratic Arithmetic Program, and is currently the most efficient representation of arithmetic circuits for showing their validity.

no known zkSNARKs that enjoy both of these properties out-of-the-box. Using transformations described in [5, 127], one can obviously lift a universal zkSNARK to be simulation-extractable, but such lift comes with either loosing succinctness, or an inevitable efficiency loss. This is true for updatable zkSNARKs as well. Namely, there is no known (out-of-the-box) simulation-extractable updatable zkSNARKs. Further, there are no transformations that can take a simulation extractable zkSNARK and make it updatable (although transformation in [5] preserves updatability).

In this paper we show that a wide class of (computationally) special-sound proofs of knowledge which have unique response property and are zero-knowledge without requiring the simulator to use the SRS trapdoor are (non-black-box) simulation-extractable when made non-interactive by the Fiat–Shamir transform. We prove that three efficient updatable universal zkSNARKs—Plonk [91], Sonic [132] and Marlin [62]—meet these requirements and conclude by showing their simulation-extractability.

Bibliography information

Chaya Ganesh, Hamidreza Khoshakhlagh, Markulf Kohlweiss, Anca Nitulescu, and Michal Zajac. What makes fiat–shamir zksnarks (updatable srs) simulation extractable? Cryptology ePrint Archive, Report 2021/511, 2021. https://eprint.iacr.org/2021/511

2.3 Chapter 5: Encryption to the Future: A Paradigm for Sending Secret Messages to Future (Anonymous) Committees

Here, we summarize our results on studying a notion of *encryption to the future*, presented in [49].

Traditional cryptographic protocols rely on secure channels between parties with publicly known identities. While knowing parties' identities can be advantageous in some aspects, it allows a mobile adversary to block parties as soon as they execute the protocol. To capture this setting, the authors in [102] proposed a model, the so-called *you only speak once* (YOSO) model, in which the adversary has strong denial of service (DoS) capabilities. The model aims to capture a setting where the adversary monitors a communication network between a large number of *one-time* stateless parties—so-called *roles* in [102]—and mounts a DoS attack on a (role-assigned) *machine* as soon as it sends a message.

An interesting technical problem to deal with in the YOSO model is to find a way to map random unknown machines M to roles R in the protocol. A role can be "party 7 in round 42 of protocol X" being executed by some yet unknown machine. If it was always known that one particular machine executes a given role it makes it easy to mount a denial of service attack just before the role is to be executed. Second, if some future role in the protocol wants to send a secret message to the machine M which executes a future role R, then we somehow need a public key of M to become known without the identity of M becoming known. The aspect of how to make it possible to send a message to a future role without anyone knowing which physical machine is going to execute the role is called the role assignment (RA) aspect.

Limitations of Prior Approaches. Existing solutions for role assignment are all unsatisfactory. One trivial solution is to use a generic witness encryption scheme [98] for an NP relation that defines the role assignment. However, existing witness encryption schemes are either based on strong assumptions such as multilinear maps [98, 100], or rely on heavy tools like indistinguishability obfuscation (iO) [97] which are currently not well understood.

In the case of RA with *short future horizon*, simpler (committee-based) solutions can be based on the approach proposed in [32, 101] over a public blockchain where a nominated set of parties, called *committee members*, assist in the role assignment process. For RA with long *future horizon*, [112] constructed a flavour of witness encryption on top of a blockchain where successive committees carry the ciphertext into the future.

The main issue with earlier approaches is the use of auxiliary committees which can add a significant communication overhead. This is often undesired in particular in a blockchain setting where resources are scarce.

Our Solutions. We investigate the problem of role assignment and study how to encrypt to a future role with strong security properties. Towards this goal, we first introduce a new primitive called *Encryption to the Future* (EtF) in the context of an underlying blockchain, where the encryption is towards parties selected at arbitrary points in the future. We show the strength of general EtF (long future horizon) as a primitive by proving that it implies a flavour of witness encryption for NP. We then weaken the definition and allow *limited* access to auxiliary committees on the blockchain such that committees transfer only a *very small* state. In this setting, we propose constructions of EtF based on a weaker primitive called *Encryption to the Current Winner* (ECW) for the setting with short future horizon. An ECW is defined as a relaxation of EtF, where the receiver of the message is determined according to the *current* state of the blockchain, i.e., selecting the receiver is based on a distribution that is known at the time of encryption. We give a construction of ECW based on standard assumptions (Oblivious Transfer and Privacy-free Garbled Circuits) that improves over the previous results [32, 101, 102] and then show our transformation that compiles any ECW into an EtF when given access to an auxiliary committee.

Bibliography information

Matteo Campanelli, Bernardo David, Hamidreza Khoshakhlagh, Anders Konring, and Jesper Buus Nielsen. Encryption to the future: A paradigm for sending secret messages to future (anonymous) committees. Cryptology ePrint Archive, Report 2021/1423, 2021. `https://eprint.iacr.org/2021/1423`

2.4 Chapter 6: (Commit-and-Prove) Predictable Arguments with Privacy

In this section, we summarize our results on privacy-preserving predictable arguments presented in [124].

The main goal of this work is to study predictable arguments (PA) [79] and witness encryption (WE) [98] schemes with privacy properties, namely zero-knowledge and witness-indistinguishability. A predictable arguments (PA) is a type of private-coin argument system

where the answer of the prover can be predicted by the verifier. That is, given the honest verifier's (private) random coins, one can compute the prover's answers in advance. The prover in such arguments is deterministic and to convince the verifier, they must be consistent with the "unique" accepting transcript throughout the entire protocol. Surprisingly, the notion of predictable arguments is closely related (and in fact equivalent) to witness encryption schemes. To build PA from WE, one can encrypt a random string r using the WE scheme and ask the prover to return r. Vice versa, one can build WE from PA by xor-ing the (predictable) answer with the plaintext.

Motivated by the above discussion, we study these two notions with privacy properties. In the context of witness encryption, such schemes are useful for applications where in addition to standard properties, one also requires some level of privacy for the decrypting party who owns a valid witness for the statement used in the encryption process. We demonstrate the usefulness of WE schemes with such decryption privacy properties by showing their application in dark pools and over-the-counter markets in which an investor (who plays the role of encrypting party here) is interested to communicate with only those trading parties (potential decrypting parties) whose financial conditions satisfy some constraint. Compared to the existing construction of zero-knowledge PA in [79], we give more efficient construction for a limited class of NP languages, namely *linear languages*. Next, we answer the open problem of constructing witness indistinguishable (WI) predictable argument in the plain model proposed in [79]. We show a transformation that converts a PA into a WI-PA still in the plain model. To that end, we use a non-interactive WI proof system as an ingredient by which the verifier convinces the prover that their challenge has been computed correctly. The prover first checks if the NIWI proof verifies and if so, computes the predicted answer as before.

Commit-and-Prove Predictable Arguments. While it has already been shown that general-purpose PA (or WE) is a powerful notion and can be used to construct several cryptographic primitives, existing constructions rely on very strong assumptions such as multilinear maps [98, 100] or indistinguishability obfuscation (iO) [97], not being yet sufficiently robust. In fact, there are evidences that assert unlikeliness of constructing strong versions of these primitives for all NP languages (yet alone schemes that provides privacy as well) [3].

To deal with the difficulty of building these primitives generally, we propose an alternative weaker notion of predictability called *Commit-and-Prove Predictable Argument* (CPPA), where except the first message of the prover, all the prover's responses can be predicted. While it is not hard to show that this notion is pretty much equivalent to the relaxed notion of WE (called cWE) in Chapter 5, here we are only interested in CPPAs with privacy guarantees, whereas in Chapter 5, privacy of sender's input is not a requirement and in some sense is an overkill for the concerned applications. We give a construction of efficient zero-knowledge CPPAs in the non-programmable random oracle model for the class of all polynomial-size circuits based on the three-round zero-knowledge argument of [92].

Bibliography information

Hamidreza Khoshakhlagh. (commit-and-prove) predictable arguments with privacy. *IACR Cryptol. ePrint Arch.*, page 377, 2022. URL `https://eprint.iacr.org/2022/377`

[3] For example, the seminal work of Garg et al. [98] shows that the existence of statistically-sound variant of these primitives for an NP-complete language implies that $\mathbf{NP} \subseteq \mathsf{AM} \cup \text{co-}\mathsf{AM}$, in turn implying the collapse of the polynomial hierarchy.

2.5 Other Contributions

Apart from the aforementioned contributions described as the main content of this thesis, I have also worked on the following results.

2.5.1 SNARK-related Projects

- In [46, 47], we model and construct a new primitive called *Succinct Publicly-Certifiable* (SPuC) proof systems, in which a party can prove knowledge of a witness by publishing a partial (designated-verifier) proof; the latter can then be certified by a committee sharing a secret in a non-interactive manner so that any party in the system can now publicly verify the proof through its certificates. Our model has practical applications in blockchains where there exist committees sharing a secret and parties are required to prove knowledge of a solution to some puzzle. The model can be also seen as a compiler that transforms designated-verifier SNARKs into succinct proof systems with a flavor of public-verifiability.

- In [50], we investigate theoretical barriers for succinct non-interactive arguments (SNARGs) by showing new negative and positive results related to extractability and to the preprocessing model. In particular, we first show the impossibility of having black-box SNARKs in the adaptive setting in the standard model. Next, we give positive results for the same question but in the non-adaptive setting by showing black-box SNARKs for a subset of NP, namely FewP, wherein statements have at most a polynomial number of valid witnesses. Lastly, we extend the Gentry-Wichs result to the SNARGs in the preprocessing model.

2.5.2 Smooth Projective Hash Functions

- In [6, 7], we study smooth projective hash functions (SPHFs) in the subversion setting. We define and construct smooth zero-knowledge hash functions (SZKHFs) as SPHFs where completeness property holds even in the case of maliciously generated language parameters and projection key.

2.6 General Preliminaries and Notation

We introduce most of the notation used in this thesis in each relevant section where it is used for the first time. Here, we only list some general notation that we use in all chapters.

2.6.1 General Notation

For any positive integer n, $[n]$ denotes the set $\{1, \ldots, n\}$. We use λ to denote the security parameter. We write $f(\lambda) \approx_\lambda g(\lambda)$ if the difference between f and g is negligible in λ. We use DPT (resp. PPT) to mean a deterministic (resp. probabilistic) polynomial time algorithm. We denote by $Y \leftarrow\!\!\$\ \mathsf{F}(X)$ a probabilistic algorithm F that on input X outputs Y. Similarly, notation $Y \leftarrow \mathsf{F}(X)$ is used for a deterministic algorithm with input X and output Y. All adversaries will be stateful. Throughout the paper, $\mathbb{F}_q$ will denote the field with q elements.

2.6.2 Bilinear groups

A bilinear group generator $\mathcal{G}(1^\lambda)$ returns public parameters $\mathsf{pp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2)$, where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are additive cyclic groups of prime order $p = 2^{\Omega(\lambda)}$, $[1]_1 = g_1, [1]_2 = g_2$ are generators of $\mathbb{G}_1$, $\mathbb{G}_2$, resp., and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate PPT-computable bilinear pairing. Depending on which notation suits better to the context, we use different notation for pairing operation. Specifically, we use multiplicative notation in Chapter 3, where $\forall U \in \mathbb{G}_1, \forall V \in \mathbb{G}_2, \forall a, b \in \mathbb{Z} : \hat{e}(U^a, V^b) = \hat{e}(U, V)^{ab}$, and additive (bracket) notation in Chapters 4 and 6, where we write $[a]_\iota$ to denote $a\,[1]_\iota$ and $\hat{e}([a]_1, [b]_2) = [a]_1 \bullet [b]_2 = [ab]_T$. The same notation naturally extends to matrices $[M]_t$ for $M \in \mathbb{Z}_p^{n \times m}$. Since every algorithm $\mathcal{A}$ takes as input the public parameters we skip them when describing $\mathcal{A}$'s input. Similarly, we do not explicitly state that each protocol starts by running $\mathcal{G}$.

Part II

Publications

Chapter 3

Balancing Privacy and Accountability in Blockchain Identity Management

In this chapter we present our results on balancing privacy and accountability in blockchain identity management. The contents of this chapter are taken almost verbatim from [70, 71], where we first presented these results.

3.1 Introduction

Early applications of blockchain to payment systems such as Bitcoin do not guarantee privacy. In the Bitcoin blockchain, blocks posted on the public ledger consist of transactions, making Bitcoin transparent – transactions are there for everybody to see. However, the identities are pseudonymous, and not tied to real world identities. Consequently, Bitcoin has the property that while the ownership of money is implicitly anonymous, the flow of money is globally visible. While this was perceived to be truly anonymous early on, there have been several works that deanonymize Bitcoin flow by analysing the payment graph [134]. To overcome the problem of lack of privacy in the first generation of cryptocurrencies such as Bitcoin, Ethereum, etc., new systems were designed to guarantee transaction privacy and anonymity for their users [25, 82, 144]. Systems like Zerocash [25] fully hide both the value inside a transaction, and the sender and receiver identities. Most blockchain use-cases, however, are hindered by complete privacy as they need *accountability* and *identity management*. Privacy-preserving systems like ZCash are not designed with accountability in mind [1]. In order to conform with regulations like “Know your customer" (KYC) and “Anti-money laundering" (AML), a legal authority should be able to learn the value and identities of the parties involved in any transaction; this requirement seems to be at odds with privacy. The seemingly contradictory requirements of transaction privacy & user anonymity, and regulatory requirements such as KYC/AML imposed on financial and banking institutions is a major hurdle in widespread adoption of the blockchain.

Our Contribution. In this work, we address the problem of balancing accountability with privacy in blockchain-based systems. We propose a new architectural design of an “identity layer" that will provide privacy for its users – that is, no one, observing the network transactions and the

[1] Zcash considers solutions to implement AML and KYC controls [1], however this solution requires trust on a single party.

status of the Blockchain should be able to learn about the identity of the owner of any account in the system. At the same time, the identity layer achieves accountability in the sense that in the presence of a reasonable suspicion, law-enforcement agencies (or other authorized parties), will be able to access the transaction history of a given user and/or block its funds, in a way similar to what is guaranteed today by traditional financial institutions. We develop cryptographic mechanisms that enhance accountability measures against misuse of the blockchain, while still providing privacy. Towards this end, we employ cryptographic techniques to design provably secure protocols, with both privacy and accountability guarantees. We prove the security of our constructions in the Universal Composability (UC) framework. We provide a high-level overview of the design of the system, and then discuss the techniques and cryptographic tools used. We believe that such an identity layer design will make Blockchain and cryptocurrencies more attractive for regulators, public institutions and traditional businesses which are interested in complying with existing legislation. In fact, the identity layer of Concordium[2], an upcoming major Blockchain project, is based on the design presented in this paper.

Overview of the System. In the proposed system, the identity and credentials of each participant in the network are initially verified and stored by authorized parties called *Identity Providers* (IPs). Each user can open a limited number of accounts where an account has an identifier that is derived from a PRF applied to a value that is between 1 and the maximal number of accounts, say n. The PRF key K is held by the user. When a user registers with an IP, K is encrypted through a threshold encryption scheme, and this ciphertext is stored with the IP. This is set up such that an appropriate number of *Anonymity Revokers* (ARs) would be able to decrypt. Standard anonymous credentials are used to certify additional attributes of the user.

When a user creates an account, they prepare some data to be published on the blockchain. This includes a threshold encryption of the account holder's public key (that was also stored with the IP at registration time). It also includes zero-knowledge proofs that the attributes the user chooses to publish in the account have been signed by the IP, and that the account identifier has been correctly computed. Thus, an account may contain complete identification of the account holder, if the user chooses to include it, or it may reveal less information, for instance the citizenship and age of the account holder.

Finally, an account includes various account specific public keys. Using the corresponding secret keys, the account holder can then perform transactions anonymously in the network. Depending on the key material included in accounts, several different ways to do transactions can be realized – this is a problem orthogonal to that of implementing the identity layer, and we give some informal examples of how this could be done in Section 3.8.

If it is suspected that an account is used for fraudulent purposes, the encrypted account information can be decrypted by a qualified set of the ARs, and an anonymous account can be linked (via the public key) to an id provided by the IP. On the other hand, if a particular user is suspected of fraud, the IP can provide its record for this user, and a qualified set of ARs can decrypt the information to learn the PRF key K. Now, they can generate the set of all values $\mathsf{PRF}_{\mathsf{K}}(x)$ for $x = 1, \ldots, n$ which are all the possible values for an account identifier. One can then identify all accounts of the user by searching the blockchain for accounts with these identifiers. Privacy is therefore guaranteed for all users, except those whose anonymity is revoked by a sufficient number of ARs.

[2] concordium.com

Note that the above also implies that we have a mechanism for preventing a user from opening an unbounded number of accounts using a single certificate from the IP: if this were possible, it would open the door for attacks where an individual registers with an IP and then allow other individuals to open accounts in their name, perhaps after payment of a small sum of money. On the other hand, we do not want the account holder to have to interact with the IP for every new account it wants to create, as this would affect efficiency. While the concrete number of accounts allowed per user is an implementation dependent parameter, our technique allows to achieve a reasonable tradeoff. The zero-knowledge proofs force the user to compute the account id's correctly, which only allows n different id's. Thus, if one attempts to open more accounts than allowed, this must result in a pre-existing account identifier, and the Blockchain will reject it.

We prove security of the system when either any number of account holders are actively corrupt, or when the identity providers are semi-honest corrupt. Note that, similar to certification authorities in standard PKI, we need some trust in the IPs: a malicious identity provider (equivalently, a malicious account holder colluding with a semi-honest IP and therefore learning the key), could produce certificates containing false identities, therefore undermining the system. Finally, depending on which properties we want to emphasize, we could tolerate different corruption levels among the anonymity revokers. Thus our system is secure in the presence of actively corrupt users and a threshold number of passively corrupt anonymity revokers; or, in the presence of passively corrupt identity-provider and a threshold number of passively corrupt anonymity revokers. In our design it is paramount that the service provided by the anonymity revokers to be available, and we want to emphasize privacy. Thus, we opt for assuming a majority of semi-honest ARs. Using standard methods, we could instead tolerate a minority of actively corrupted ARs.

Overview of technical ideas. We use cryptographic schemes such as Pedersen commitments [143], Dodis-Yampolskiy PRF [77], Pointcheval-Sanders (PS) signature scheme [145], CL encryption scheme [54]. We use zkSNARKs in combination with commitments and signatures in the spirit of [9, 48]: the PS blind signature we use is defined using groups of a certain prime order, and when a user proves knowledge of a signature, the message that is signed is committed to using a Pedersen-type commitment in such a group. Now, we can use standard sigma-protocols to provide commitments to individual attributes of the user in the same group, and finally use SNARKs on committed messages to show statements such as "the age attribute of the user is a number greater than 18". In this way, we only need to use SNARKs on rather small circuits, and we can achieve much greater efficiency than if we had to convert large statements involving, e.g, group operations into a Boolean circuit to be evaluated inside the SNARK. In this way, creating an account requires a constant number of exponentiations (i.e., independent of the security parameter), and likewise, the number of group elements in an account is constant.

We provide a generic lifting transformation for Fiat Shamir NIZKs for DL-languages into UC NIZKs. While such a transformation by encrypting the witness under a key that is part of the CRS (and the secret key part of the CRS trapdoor) is folklore [75], using the CL encryption scheme allows us to efficiently prove statements about values in the exponent, which is novel to the best of our knowledge.

Related Work. The cryptographic tools used in building our solution, like commitment schemes, blind signatures, zero-knowledge proofs, and threshold encryption are based on anonymous credentials technology. Anonymous credentials [60] allow a party to prove to a

verifier that one has a set of credentials without revealing anything beyond this fact. Revocable anonymity [43, 125] allows a trusted third party to discover the identity of all otherwise anonymous participants. Conditional anonymity requires that a user's transactions remain anonymous until certain conditions are violated [44, 45, 69]. In [69], an unclonable identification scheme is introduced, that is, roughly, an identification scheme where honest users can identify themselves anonymously as members of a group, but where clones of users can be detected and have their identities revealed if they identify themselves simultaneously. This was extended from one-time authentication to n-times anonymous authentication in [45] where a certain number of unlinkable accounts are derived that can later be efficiently traced. The works of [13, 139, 154, 155] addressed related problems of allowing a user to show a credential anonymously and unlinkably up to n times to a particular verifier. The potential for abuse of unconditional anonymity by misbehaving users has been articulated in the context of group signatures. In a group signature scheme, each group member can sign a message on behalf of a group such that anyone can verify that the group signature is produced by someone in the group, but not who exactly. Our idea for identifying all accounts of a user in case of revocation by using a PRF to generate account identifiers is reminiscent of the work of traceable signatures [63] that enable a tracing agent to identify all signatures produced by a particular member. The idea of deriving a certain number of unlinkable accounts that can later be efficiently traced has been used in various forms in the anonymous credential literature [14, 45, 63] for the purposes of balancing accountability and anonymity.

Unfortunately none of the previous works seem to fit our intended use case, which motivated us to design the system described in this paper. Moreover, the toolbox of efficient tools available to the protocol designer has grown in recent years (e.g., the CL encryption scheme, advances in SNARKs, etc.), which also motivates exploring new designs.

The zkLedger protocol [137] is an asset transfer scheme that hides transaction amounts and sender-receiver relationship, and supports auditing. The protocol is for a setting where the transacting parties are banks, and requires the participation of the banks for an audit to take place. The work of [10] presents a privacy-preserving token management system that supports auditing in permissioned blockchains. The system of [10] is in the UTXO framework, where users own tokens that are certified, and prove ownership of tokens in a privacy-preserving manner. In contrast, we work in the account-based model; and our design is modular – the identity layer is separate from the transaction layer. One main difference of our work from the works of [137] and [10] is that while both these works assume that the entire system is permissioned, again our design is more modular: our ID layer obviously assumes that IPs and ARs are known (and trusted to some extent) and is therefore in some sense permissioned. However, the ID layer can work on top of the consensus mechanism of a permissionless blockchain, i.e., any blockchain that can be abstracted using the ledger functionality $\mathcal{F}_{\mathsf{ledger}}$.

Finally, Solidus [57] is a privacy-preserving system that allows customers of financial institutions (e.g., banks) to transfer assets and ensures that only the banks of the sender and receiver can learn the transaction details. While there is no explicit audit functionality in Solidus, banks can reveal the content of a suspicious transaction to the authorized auditors. However this approach requires to trust a single party (i.e., the bank).

3.2 Preliminaries and Building Blocks

This section defines our notation and introduces the cryptographic schemes we use in our construction.

3.2.1 Notation

we refer the reader to Section 2.6 for general notation. We use the identifier AH for account holder, IP for identity provider and AR for anonymity revoker. By an identifier, we mean an arbitrary string that uniquely identifies a party.

3.2.2 Pseudorandom Functions

We recall the standard notion of pseudorandom functions.

Definition 3.2.1 (PRF)**.** *Let* $\mathsf{PRF}\colon \mathcal{K} \times \mathsf{D} \to \mathsf{R}$ *be a family of functions and let* Γ *be the set of all functions* $\mathsf{D} \to \mathsf{R}$*. We say that* PRF *is a pseudorandom function (PRF) (family) if it is efficiently computable and for all PPT distinguishers* $\mathcal{D}$

$$\left|\Pr\left[\mathsf{K} \leftarrow\!\$\, \mathcal{K}, \mathcal{D}^{\mathsf{PRF}_{\mathsf{K}}(\cdot)}(1^\lambda)\right] - \Pr\left[g \leftarrow\!\$\, \Gamma, \mathcal{D}^{g(\cdot)}(1^\lambda)\right]\right| \approx_\lambda 0.$$

We define a weak notion of PRF robustness, meaning that is should be hard to find a key that produce collisions with the PRF evaluation of an honest user. Our definition is similar to the one in [80], but here one of the two keys is chosen honestly.

Definition 3.2.2 (Weakly Robust PRF)**.** *A PRF is weakly robust if:*

$$\Pr[\mathsf{K} \leftarrow\!\$\, \mathsf{Gen}(1^\lambda), (x^*, \mathsf{K}^*) \leftarrow\!\$\, \mathcal{A}^{\mathsf{PRF}_{\mathsf{K}}(\cdot)}(1^\lambda) : \exists (x, y) \in \mathcal{Q}, \mathsf{PRF}_{\mathsf{K}^*}(x^*) = y] \approx_\lambda 0$$

where $\mathcal{Q}$ *is the set of inputs/outputs of the oracle available to the adversary.*

Instantiation with Dodis-Yampolskiy PRF.

We use the PRF of Dodis and Yampolskiy [77] that operates in a group $\mathbb{G}$ of order p with generator g. On input x and the PRF key $\mathsf{K} \leftarrow\!\$\, \mathbb{F}_p$, $\mathsf{PRF}_{\mathsf{K}}(x) = g^{1/\mathsf{K}+x}$. This is shown to be pseudorandom under the Decisional Diffie-Hellman Inversion assumption in group $\mathbb{G}$. Note that the security holds only for small domains, namely inputs that are slightly superlogarithmic in the security parameter, but this is sufficient for our work, as the maximum number of accounts a user can open is less than a constant $\mathsf{Max}_{\mathsf{ACC}}$. It can also be easily shown that the Dodis-Yampolskiy PRF is weakly robust: using the PRF assumption, we can replace the output of the PRF oracle with random group elements. If the adversary outputs an input x^* and key K^* that are compatible with one of the output of the oracles, we can compute the discrete logarithm of that element as $1/(\mathsf{K}^* + x^*)$.

3.2.3 Commitment Schemes

Definition 3.2.3 (Commitment Schemes)**.** *A commitment scheme* $\mathbf{Com} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{OpenVrf})$ *consists of the following algorithms:*

- $\mathit{Setup}(1^\lambda)$*: given the security parameter* 1^λ*, it outputs a commitment key* ck*. The commitment key* ck *defines a message space* $\mathcal{S}_m$ *and a randomizer space* $\mathcal{S}_r$*.*

- $\mathit{Commit}_{\mathsf{ck}}(m)$*: a probabilistic algorithm that given a message* m*, outputs a pair* $(cm, r_{cm}) \leftarrow\$ \mathit{Commit}_{\mathsf{ck}}(m)$ *of commitment* cm *and an opening* r_{cm}*. We sometimes write* $\mathit{Commit}_{\mathsf{ck}}(m; r_{cm})$ *when we want to be able to fix the value of the randomness* r_{cm} *to a specific value.*

- $\mathit{OpenVrf}_{\mathsf{ck}}(cm, m, r_{cm})$*: a deterministic algorithm that outputs a bit indicating acceptance or rejection.*

Hiding property. We say that **Com** is hiding if for all non-uniform PPT adversary $\mathcal{A}$, there exists a negligible function $negl(\lambda)$ such that

$$\left| \frac{1}{2} - \Pr\left[b' = b \;\middle|\; \begin{array}{c} \mathsf{ck} \leftarrow\$ \mathsf{Setup}(1^\lambda); (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{ck}); \\ \\ b \leftarrow\$ \{0,1\}; \quad (cm, r_{cm}) \leftarrow\$ \mathsf{Commit}_{\mathsf{ck}}(m_b); \quad b' \leftarrow\$ \mathcal{A}(cm); \end{array} \right] \right| \leq negl(\lambda)$$

where the probability is over the randomness of $\mathcal{A}$, Setup, Commit and the choice of bit b. We say that **Com** is perfectly hiding if $negl(\lambda) = 0$.

Binding property. A commitment scheme **Com** is binding if for all non-uniform PPT adversary $\mathcal{A}$, there exists a negligible function $negl(\lambda)$ such that

$$\Pr\left[\begin{array}{c} \mathsf{OpenVrf}_{\mathsf{ck}}(cm, m_0, r_0) = 1 \wedge \\ \mathsf{OpenVrf}_{\mathsf{ck}}(cm, m_1, r_1) = 1 \wedge m_0 \neq m_1 \end{array} \;\middle|\; \begin{array}{c} \mathsf{ck} \leftarrow\$ \mathsf{Setup}(1^\lambda); \\ \\ \\ (cm, m_0, m_1, r_0, r_1) \leftarrow\$ \mathcal{A}(\mathsf{ck}); \end{array} \right] \leq negl(\lambda)$$

where the probability is over the randomness of $\mathcal{A}$ and Setup. We say that **Com** is perfectly binding if $negl(\lambda) = 0$.

3.2.4 Blind Signature Schemes

We adapt the notation of [152] to two-round blind signature schemes.

Definition 3.2.4 (Blind Signature Schemes)**.** *An interactive signature scheme between a signer* S *and user* U *consists of a tuple of efficient algorithms* $\mathbf{BS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}_1, \mathsf{Sign}_2, \mathsf{Unblind}, \mathsf{VerifySig})$ *where*

- $\mathsf{Setup}(1^\lambda)$*, on input the security parameter* 1^λ *outputs* pp*, which is given implicitly as input to all other algorithms, even when omitted.*

- $\mathsf{KeyGen}(\mathsf{pp})$*, on input the public parameter* pp *generates a key pair* $(\mathsf{sk}, \mathsf{pk})$ *for security parameter* λ*.*

- $\mathsf{Sign}_1(\mathsf{pk}, m)$*, which is run by* U*, takes as input* pk *and a message* $m \in \{0,1\}^*$ *and outputs* $\mathtt{sign}_1$ *and* ω *(w.l.o.g.* ω *can be thought of as the randomness used to run* Sign_1*).*

- $\mathsf{Sign}_2(\mathsf{sk}, \mathtt{sign}_1)$*, which is run by* S*, takes as input* sk *and* $\mathtt{sign}_1$ *and outputs* $\mathtt{sign}_2$*.*

- $\mathsf{Unblind}(\mathtt{sign}_2, \omega)$*, which is run by* U*, takes as input* $\mathtt{sign}_2, \omega$ *and outputs* σ*.*

$\mathsf{Exp}^{\mathsf{Blind}}_{\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow\$ \mathsf{Setup}(1^\lambda); (\mathsf{sk}, \mathsf{pk}) \leftarrow\$ \mathsf{KeyGen}(\mathsf{pp});$

$(m_0, m_1) \leftarrow \mathcal{A}(\mathsf{sk}, \mathsf{pp});$

$b \leftarrow\$ \{0,1\};$

$(\mathtt{sign}_1, \omega) \leftarrow\$ \mathsf{Sign}_1(\mathsf{pk}, m_b);$

$(\overline{\mathtt{sign}}_1, \overline{\omega}) \leftarrow\$ \mathsf{Sign}_1(\mathsf{pk}, m_{1-b});$

$\mathtt{sign}_2, \overline{\mathtt{sign}}_2 \leftarrow \mathcal{A}(\mathtt{sign}_1, \overline{\mathtt{sign}}_1);$

let σ_b, σ_{1-b} denote the (possibly undefined) local outputs of

$\mathsf{Unblind}(\mathtt{sign}_2, \omega)$ and $\mathsf{Unblind}(\overline{\mathtt{sign}}_2, \overline{\omega})$ respectively.

if $\mathsf{VerifySig}(\mathsf{pk}, m_0, \sigma_0) = 0$ or $\mathsf{VerifySig}(\mathsf{pk}, m_1, \sigma_1) = 0$, **then** $(\sigma_0, \sigma_1) = (\perp, \perp);$

$b^* \leftarrow \mathcal{A}(\sigma_0, \sigma_1);$

if $b = b^* \wedge |m_0| = |m_1|$, **then return** 1; **else return** 0;

Figure 3.1: Experiment in the definition of blindness property

- $\mathsf{VerifySig}(\mathsf{pk}, m, \sigma)$ *outputs a bit.*

Remark 3.2.5. Note that a blind signature scheme implicitly defines a normal signature scheme as well, where the signing algorithm $\mathsf{Sign}(\mathsf{sk}, m)$ simply emulates a blind signature protocol and outputs the resulting signature σ.

The correctness property of the scheme requires that the following holds: for any $(\mathtt{pp}) \leftarrow\$ \mathsf{Setup}(1^\lambda)$, $(\mathsf{sk}, \mathsf{pk}) \leftarrow\$ \mathsf{KeyGen}(\mathtt{pp})$, any message $m \in \{0,1\}^*$, if $(\mathtt{sign}_1, \omega) \leftarrow\$ \mathsf{Sign}_1(\mathsf{pk}, m)$, $\mathtt{sign}_2 \leftarrow\$ \mathsf{Sign}_2(\mathsf{sk}, \mathtt{sign}_1)$, $\sigma = \mathsf{Unblind}(\mathtt{sign}_2, \omega)$ then $\mathsf{VerifySig}(\mathsf{pk}, m, \sigma) = 1$ with overwhelming probability over $\lambda \in \mathbb{N}$.

We require the standard notion of *existential unforgeability under chosen message attacks* (EUF-CMA) [108]. The blind signature scheme we use should additionally satisfy two properties, namely *blindness* and *simulatability*. Blindness captures the requirement that we run two blind signing protocols on two messages of the adversary's choice, and the adversary should not be able to say which input-signature pair corresponds to which execution. This is stronger than the definition in [145]; however, we can show that the PS blind signature scheme can be made to satisfy this stronger, standard definition. This follows from the rerandomizability of PS signatures.

Definition 3.2.6 (Blindness)**.** *An interactive signature scheme* $\mathbf{BS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}_1, \mathsf{Sign}_2, \mathsf{Unblind}, \mathsf{VerifySig})$ *is called blind if for any PPT adversary* $\mathcal{A}$, $\Pr[\mathsf{Exp}^{\mathsf{Blind}}_{\mathcal{A}}(\lambda) = 1] \approx_\lambda 1/2$, *where the experiment* $\mathsf{Exp}^{\mathsf{Blind}}_{\mathcal{A}}(\lambda)$ *is defined in Fig. 3.1.*

Definition 3.2.7 (Simulatability)**.** *An interactive signature scheme* $\mathbf{BS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}_1, \mathsf{Sign}_2, \mathsf{Unblind}, \mathsf{VerifySig})$ *is called simulatable if there exist a PPT algorithm* $\mathcal{S}$ *s.t. for any PPT adversary* $\mathcal{A}$, $\Pr[\mathsf{Exp}^{\mathsf{Sim}}_{\mathcal{A}}(\lambda) = 1] \approx_\lambda 1/2$, *where the experiment* $\mathsf{Exp}^{\mathsf{Sim}}_{\mathcal{A}}(\lambda)$ *is defined in Fig. 3.2.*

$\mathsf{Exp}^{\mathsf{Sim}}_{\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow\!\$\ \mathsf{Setup}(1^\lambda); (\mathsf{sk}, \mathsf{pk}) \leftarrow\!\$\ \mathsf{KeyGen}(\mathsf{pp});$

$b \leftarrow\!\$\ \{0,1\};$

$(m; r) \leftarrow \mathcal{A}(\mathsf{sk}, \mathsf{pp});$

$(\mathtt{sign}_1, \omega) \leftarrow\!\$\ \mathsf{Sign}_1(\mathsf{pk}, m; r);$

$\mathtt{sign}_2^0 \leftarrow\!\$\ \mathsf{Sign}_2(\mathsf{sk}, \mathtt{sign}_1);$

$\sigma = \mathsf{Unblind}(\mathtt{sign}_2^0, \omega);$

$\mathtt{sign}_2^1 \leftarrow\!\$\ \mathcal{S}(\mathsf{pk}, m, \omega, \sigma);$

$b^* \leftarrow \mathcal{A}(\sigma, m, \mathtt{sign}_2^b);$

if $b = b^*$, **then return** 1; **else return** 0;

Figure 3.2: Experiment in the definition of simulatability property

Instantiation with PS Signature scheme [145]

We specify here the multi-message Pointcheval-Sanders (PS) signature scheme and how it realizes our definition of Blind signature. The PS scheme is a randomizable signature that uses groups with asymmetric pairing and allows to sign messages that are completely unknown to the signer.

- $\mathsf{Setup}(1^\lambda)$: Sample $\mathsf{pp} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, g_2) \leftarrow\!\$\ \mathcal{G}(1^\lambda)$.
- $\mathsf{KeyGen}(\mathsf{pp})$: Given $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, g_2)$, the key generation works as follows:
  1. Choose $x, y_i \leftarrow\!\$\ \mathbb{F}_p$ for $i = 1, \ldots, \ell$[3].
  2. Set $X = g_1^x, \tilde{X} = g_2^x, Y_i = g_1^{y_i}$ and $\tilde{Y}_i = g_2^{y_i}$ for $i = 1, \ldots, \ell$.
  3. The public key is now $(\tilde{X}, \{Y_i\}_{i\in[\ell]}, \{\tilde{Y}_i\}_{i\in[\ell]})$, while the secret key is X.
- $\mathsf{Sign}_1(\mathsf{pk}, m = (m_1, \ldots, m_\ell))$: The user chooses $\omega \leftarrow\!\$\ \mathbb{F}_p$, and computes $\mathtt{sign}_1 = g_1^\omega \prod_{i=1}^{\ell} Y_i^{m_i}$. Note that ω is random and not used for anything else, and therefore $\mathtt{sign}_1$ perfectly hides the rest of the m_i.
- $\mathsf{Sign}_2(\mathsf{sk}, \mathtt{sign}_1)$: The signer chooses $\alpha \leftarrow\!\$\ \mathbb{F}_p$ and sets $a' = g_1^\alpha \neq 1_{\mathbb{G}_1}$ and $b' = (X \cdot \mathtt{sign}_1)^\alpha$. They output $\mathtt{sign}_2 = (a', b')$.
- $\mathsf{Unblind}(\mathtt{sign}_2, \omega)$: For $\mathtt{sign}_2 = (a', b')$, the user first computes $\hat{\sigma} = (a', b'/a'^\omega)$. Then, they rerandomize $\hat{\sigma}$ by choosing $\gamma \leftarrow\!\$\ \mathbb{F}_p$ and computing $\sigma = \hat{\sigma}^\gamma = (\hat{\sigma}_1^\gamma, \hat{\sigma}_2^\gamma)$. The user returns σ.
- $\mathsf{VerifySig}(\mathsf{pk}, m = (m_1, \ldots, m_\ell), \sigma)$: The algorithm parses σ as (σ_1, σ_2) and outputs 1 if the following checks pass:
  1. $\sigma_1 \neq 1_{\mathbb{G}_1}$.
  2. $\hat{e}(\sigma_1, \tilde{X} \prod \tilde{Y}_j^{m_j}) = \hat{e}(\sigma_2, g_2)$.

[3]Note that the scheme allows to sign vectors over $\mathbb{F}_p$ of length ℓ, so ℓ should be chosen with this in mind.

The PS signature satisfies Correctness, Unforgeability, Blindness and Simulatability.
Correctness. If $\sigma = (\sigma_1 = h, \sigma_2 = h^{x+\sum y_j m_j})$, then

$$\hat{e}(\sigma_1, \tilde{X}\prod \tilde{Y}_j^{m_j}) = \hat{e}(h, \tilde{X}\prod \tilde{Y}_j^{m_j}) = \hat{e}(h, g_2)^{x+\sum y_j m_j}$$
$$= \hat{e}(h^{x+\sum y_j m_j}, g_2) = \hat{e}(\sigma_2, g_2).$$

Unforgeability. As shown in [145], the EUF-CMA security of this signature is equivalent to the single-message version, where its security relies on the following assumption[4].

Definition 3.2.8 (PS Assumption 1 [145])**.** *Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, g_2)$ be a bilinear group setting of type 3. For $(X = g_1^x, Y = g_2^y)$ and $(\tilde{X} = g_2^x, \tilde{Y} = g_2^y)$, where x and y are random scalars in $\mathbb{Z}_p$, we define the oracle $\mathcal{O}(m)$ on input $m \in \mathbb{Z}_p$ that chooses a random $h \in \mathbb{G}_1$ and outputs the pair $P = (h, h^{x+my})$. Given $(g_1, Y, g_2, \tilde{X}, \tilde{Y})$ and unlimited access to this oracle, no adversary can efficiently generate such a pair, with $h \neq 1_{\mathbb{G}_1}$, for a new scalar m^*, not asked to $\mathcal{O}$.*

Blindness. Let us define a a game H_0 which is define as game $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Blind}}$ with the only difference that $b = 0$. We can also define a symmetrical experiment H_1 where the bit $b = 1$.

To claim indistinguishability between this two games we make the following observations: 1) $\mathtt{sign}_1$ perfectly hides m; 2) the message $\mathtt{sign}_2$ coming from a corrupted signer is rerandomized (thought Unblind algorithm) before the final signature σ is given to adversary $\mathcal{A}$. From the above two observations follow that the execution in which $\mathcal{A}$ participates first in the computation of σ_0 and then of σ_1 is identically distributed to the one where $\mathcal{A}$ participate first in the computation of σ_1 and then of σ_0. Therefore H_0 is indistinguishable from H_1.

Simulatability. We start by defining the simulator $\mathcal{S}$. $\mathcal{S}$ given m, ω and the signature $\sigma = (\sigma_1, \sigma_2)$ chooses $\beta \leftarrow\!\!\$\ \mathbb{F}_p$ and returns $\mathtt{sign}_2 = (a', b')$ where $a' = \sigma_1^{\beta}$ and $b' = \sigma_2^{\beta} \cdot \sigma_1^{\omega \cdot \beta}$.

We now argue that the game H_0 where $\mathtt{sign}_2$ is computed honestly (i.e., as output of Sign_2) is identical distribute to a simulated game H_1 where $\mathtt{sign}_2$ is computed by $\mathcal{S}$. This indistinguishability follows from the fact that distribution of the message $\mathtt{sign}_2$ output by $\mathcal{S}$ and the distribution of the output of Sign_2 are identical. We conclude the proof observing that H_0 corresponds to the experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Sim}}$ where the bit $b = 0$ and H_1 corresponds to $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Sim}}$ with $b = 1$.

3.2.5 Secret Sharing Scheme

Informally, a (n, d)-secret sharing of a secret value s is an encoding of s into n shares, such that any $\mathsf{d} + 1$ shares together can reconstruct s, whereas fewer shares reveal no information about s.

Definition 3.2.9. *A (n, d)-secret sharing scheme $\mathbf{SS} = (\mathit{Share}, \mathit{Reconstruct})$ over message space S consists of the following algorithms:*

- $\mathit{Share}^{\mathsf{n},\mathsf{d}}(s; r) \rightarrow ([s]_1, \ldots, [s]_{\mathsf{n}})$ *is a randomized algorithm that on any input $s \in S$ and randomness r, outputs n shares $([s]_1, \ldots, [s]_{\mathsf{n}})$.*
- $\mathit{Reconstruct}^{\mathsf{n},\mathsf{d}}([s]_{i_1}, \ldots, [s]_{i_{\mathsf{d}+1}}) \rightarrow s'$ *is a deterministic algorithm that takes $\mathsf{d} + 1$ shares as input and returns the reconstructed message $s' \in S$.*

[4]The assumption is related to the LRSW assumption and is proved in [145] to hold in the bilinear generic group model.

$\mathsf{Exp}^{\mathsf{SS-Sim}}_{\mathcal{A}}(\lambda, \mathsf{n}, \mathsf{d})$

$(\mathcal{C}, s_0, s_1) \leftarrow \mathcal{A}(\mathsf{n}, \mathsf{d})$;

if $s_0 \notin S \vee s_1 \notin S \vee |s_0| \neq |s_1|$ **then return** 0;

$([s_0]_1, \ldots, [s_0]_{\mathsf{n}}) \leftarrow \mathsf{Share}^{\mathsf{n},\mathsf{d}}(s_0)$;

$([s_1]_1, \ldots, [s_1]_{\mathsf{n}}) \leftarrow \mathsf{Share}^{\mathsf{n},\mathsf{d}}(s_1)$;

$\{[s'_0]_i\}_{i\in[\mathsf{n}]\setminus\mathcal{C}} \leftarrow \mathsf{SimShare}(\mathsf{n}, \mathsf{d}, \{[s_1]_i\}_{i\in\mathcal{C}}, s_0)$;

$b \leftarrow \{0,1\}$; **if** $b = 0$ **then** $x = \{[s_0]_i\}_{i\in[\mathsf{n}]}$;

else $x = \{[s_1]_i\}_{i\in\mathcal{C}} \cup \{[s'_0]_i\}_{i\in[\mathsf{n}]\setminus\mathcal{C}}$;

$b' \leftarrow \mathcal{A}(x)$; **if** $b' = b \wedge |\mathcal{C}| \leq \mathsf{d}$ **then return** 1;

else return 0;

Figure 3.3: Share Simulatability experiment for Secret Sharing

Informally, correctness requires that $s' = s$, and privacy requires that given d or fewer shares of either s_0 or s_1, no efficient adversary can guess which message was shared.

Share Simulatability. We additionally use another property for a secret sharing as defined in [147] which requires that given any set of d or fewer shares of s_0 and given s_1, there exists a PPT algorithm $\mathsf{SimShare}$ that generates the rest of the shares in such a way that is indistinguishable from a fresh sharing of s_1.

Definition 3.2.10 (Share Simulatability)**.** *We say that a* (n, d)*-secret sharing scheme* $\mathbf{SS} = (\mathit{Share}, \mathit{Reconstruct})$ *over message space* S *is share simulatable if there exists an efficient simulation algorithm* $\mathit{SimShare}$ *such that for all PPT adversaries* $\mathcal{A}$*,* $\Pr[\mathsf{Exp}^{\mathsf{SS-Sim}}_{\mathcal{A}}(\lambda, \mathsf{n}, \mathsf{d}) = 1] \approx_\lambda 1/2$*, where the experiment* $\mathsf{Exp}^{\mathsf{SS-Sim}}_{\mathcal{A}}(\lambda, \mathsf{n}, \mathsf{d})$ *is described in Fig. 3.3.*

3.2.6 (Ad-hoc) Threshold Encryption Scheme

We recall the definition of an ad-hoc threshold encryption scheme here.

Definition 3.2.11. *A* (n, d)*-threshold encryption scheme* $\mathbf{TE} = (\mathsf{TKeyGen}, \mathsf{TEnc}, \mathsf{ShareDec}, \mathsf{TCombine})$ *over message space* M *consists of the following algorithms:*

- $\mathsf{TKeyGen}(1^\lambda)$ *is a randomized key generation algorithm that takes the security parameter* λ *as input and returns a private-public key pair* $(\mathsf{sk}, \mathsf{pk})$.
- $\mathsf{TEnc}^{\mathsf{n},\mathsf{d}}_{\mathbf{pk}_R}(m)$*, a probabilistic encryption algorithm that encrypts a message* $m \in M$ *to a set of public keys* $\mathbf{pk}_R = \{\mathsf{pk}_i\}_{i\in R}$ *in such a way that any size* $\mathsf{d}+1$ *subset of the recipient set should jointly be able to decrypt. We sometimes write* $\mathsf{TEnc}^{\mathsf{n},\mathsf{d}}_{\mathbf{pk}_R}(m; r)$ *when we want to be able to fix the value of the randomness* r *to a specific value.*
- $\mathsf{ShareDec}^{\mathsf{n},\mathsf{d}}_{\boldsymbol{pk}_R,\mathsf{sk}_i}(ct)$*, on input a ciphertext* ct *and a secret key* sk_i*, outputs a decryption share* μ_i.
- $\mathsf{TCombine}^{\mathsf{n},\mathsf{d}}_{\boldsymbol{pk}_R}(ct, \{\mu_i\}_{i\in I})$*, a deterministic algorithm that takes a subset* $I \subset [n]$ *with size* $\mathsf{d}+1$ *of decryption shares* $\{\mu_i\}_{i\in I}$ *and outputs either a message* $m \in M$ *or* $\perp$.

$\mathsf{Exp}^{\mathsf{SSS}}_{\mathcal{A}}(\lambda, \mathsf{n}, \mathsf{d})$

$[n] \supseteq \mathcal{C} \leftarrow \mathcal{A}(\lambda, \mathsf{n}, \mathsf{d});$

$(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow\!\!\$\ \mathsf{TKeyGen}(1^\lambda)$ **for** $i \in [n];$

$([n] \supseteq R, m_0, m_1) \leftarrow \mathcal{A}(\{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in\mathcal{C}});$

$b \leftarrow\!\!\$\ \{0,1\}; ct \leftarrow\!\!\$\ \mathsf{TEnc}^{\mathsf{n},\mathsf{d}}_{\mathbf{pk}_R}(m_b);$

$b' \leftarrow \mathcal{A}(ct);$

if $\Big(b' = b \wedge |m_0| = |m_1| \wedge |R \cap \mathcal{C}| \leq \mathsf{d}\Big);$

then return 1; **else return** $0;$

Figure 3.4: Static Semantic Security Experiment for Threshold Encryption

$\mathsf{Exp}^{\mathsf{PDS}}_{\mathcal{A}}(\lambda, \mathsf{n}, \mathsf{d})$

$[n] \supseteq \mathcal{C} \leftarrow \mathcal{A}(\lambda, \mathsf{n}, \mathsf{d});$

$(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow\!\!\$\ \mathsf{TKeyGen}(1^\lambda)$ **for** $i \in [n];$

$([n] \supseteq R, m_0, m_1) \leftarrow \mathcal{A}(\{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in\mathcal{C}});$

$b \leftarrow\!\!\$\ \{0,1\}; ct_0 \leftarrow\!\!\$\ \mathsf{TEnc}^{\mathsf{n},\mathsf{d}}_{\mathbf{pk}_R}(m_0); ct_1 \leftarrow\!\!\$\ \mathsf{TEnc}^{\mathsf{n},\mathsf{d}}_{\mathbf{pk}_R}(m_1);$

if $b = 0$ **then** $\mu_i = \mathsf{ShareDec}^{\mathsf{n},\mathsf{d}}_{\boldsymbol{pk}_R,\mathsf{sk}_i}(ct_0)$ **for** $i \in R \backslash \mathcal{C};$

else if $b = 1$ **then** $\Big(\mu_i = \mathsf{ShareDec}^{\mathsf{n},\mathsf{d}}_{\boldsymbol{pk}_R,\mathsf{sk}_i}(ct_1)$ **for** $i \in R \cap \mathcal{C}$

$\wedge\ \mu_i = \mathsf{SimPart}(\mathsf{n}, \mathsf{d}, \boldsymbol{pk}_R, c_0, \{\mu_j\}_{j\in R\cap\mathcal{C}}, m_0)$ **for** $i \in R \backslash \mathcal{C}\Big);$

$b' \leftarrow \mathcal{A}(ct_b, \{\mu_j\}_{j\in R\backslash\mathcal{C}});$

if $\Big(b' = b \wedge |m_0| = |m_1| \wedge |R \cap \mathcal{C}| \leq \mathsf{d}\Big);$

then return 1; **else return** $0;$

Figure 3.5: Partial Decryption Simulatability Experiment for Threshold Encryption

We use the static security definition of Reyzin et al. [147] for threshold encryption schemes which requires two properties, namely *static semantic security* and *partial decryption simulatability*.

Definition 3.2.12. *A* (n, d)*-threshold encryption scheme* $\mathbf{TE} = (\mathsf{TKeyGen}, \mathsf{TEnc}, \mathsf{ShareDec}, \mathsf{TCombine})$ *is* (n, d)*-statically semantic secure (SSS) if for all PPT adversaries* $\mathcal{A}$*,* $\Pr[\mathsf{Exp}^{\mathsf{SSS}}_{\mathcal{A}}(\lambda, \mathsf{n}, \mathsf{d}) = 1] \approx_\lambda 1/2$*, where the experiment* $\mathsf{Exp}^{\mathsf{SSS}}_{\mathcal{A}}(\lambda, \mathsf{n}, \mathsf{d})$ *is defined in Fig. 3.4.*

Definition 3.2.13. *We say that a* (n, d)*-threshold encryption scheme* $\mathbf{TE} = (\mathsf{TKeyGen}, \mathsf{TEnc}, \mathsf{ShareDec}, \mathsf{TCombine})$ *is* (n, d)*-partial decryption simulatable (PDS) if there exists an efficient algorithm* $\mathsf{SimPart}$ *such that for all PPT adversaries* $\mathcal{A}$*,* $\Pr[\mathsf{Exp}^{\mathsf{PDS}}_{\mathcal{A}}(\lambda, \mathsf{n}, \mathsf{d}) = 1] \approx_\lambda 1/2$*, where the experiment* $\mathsf{Exp}^{\mathsf{PDS}}_{\mathcal{A}}(\lambda, \mathsf{n}, \mathsf{d})$ *is described in Fig. 3.5.*

Definition 3.2.14. *A threshold encryption scheme* $\mathbf{TE} = (\mathsf{KeyGen}, \mathsf{TEnc}, \mathsf{ShareDec}, \mathsf{TCombine})$ *is* (n, d)*-statically secure if it is both* (n, d)*-statically semantically secure and* (n, d)*-partial decryption simulatable.*

Construction based on Share and Encrypt paradigm.

Below, we recall the construction of [74]. Let $\mathbf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption scheme and $\mathbf{SS} = (\mathsf{Share}, \mathsf{Reconstruct})$ be a secret sharing scheme.

- $\mathsf{TKeyGen}(1^\lambda)$: return $\mathsf{KeyGen}(1^\lambda)$.
- $\mathsf{TEnc}^{\mathsf{n},\mathsf{d}}_{\mathbf{pk}_R}(m)$: return $ct = (ct_1, \ldots, ct_\mathsf{n})$, where $ct_i = \mathsf{Enc}_{\mathsf{pk}_i}(m_i)$ for $i \in R$ and $(m_1, \ldots, m_\mathsf{n}) = \mathsf{Share}^{\mathsf{n},\mathsf{d}}(m)$.
- $\mathsf{ShareDec}^{\mathsf{n},\mathsf{d}}_{\boldsymbol{pk}_R,\mathsf{sk}_i}(ct)$: let $ct = (ct_1, \ldots, ct_\mathsf{n})$. Return decryption share $\mu_i = \mathsf{Dec}_{\mathsf{sk}_i}(ct_i)$.
- $\mathsf{TCombine}^{\mathsf{n},\mathsf{d}}_{\boldsymbol{pk}_R}(ct, \{\mu_i\}_{i\in I})$: Return $m = \mathsf{Reconstruct}^{\mathsf{n},\mathsf{d}}(\{\mu_i\}_{i\in I})$.

The following theorem is proved in [147] (see Appendix D, Thm. 11 of [147]).

Lemma 3.2.15. *The threshold encryption scheme* $\mathbf{TE}$ *described above is* (n, d)*-statically secure, as long as* $\mathbf{SS}$ *is a secure share simulatable* (n, d)*-secret sharing scheme, and* $\mathbf{PKE}$ *is a CPA-secure public key encryption scheme.*

3.2.7 Non-Interactive Zero-Knowledge Proofs

Definition 3.2.16 (NIZK)**.** *A non-interactive zero-knowledge proof system (NIZK) for an NP language* $\mathcal{L}$ *with relation* $\mathcal{R}_\mathcal{L}$ *consists of the following four algorithms:*

- $\mathsf{GenCRS}(1^\lambda, \mathcal{L})$. *On input* 1^λ *and the description of the language* $\mathcal{L}$*, generates a common reference string* crs*, a trapdoor* τ *and an extraction key* ek.
- $\mathsf{P}(\mathsf{crs}, \mathsf{x}, \mathsf{w})$. *On input of a* crs*, a statement* x *with witness* w*, outputs a proof* π.
- $\mathsf{V}(\mathsf{crs}, \mathsf{x}, \pi)$. *Given a* crs*, a statement* x *and a proof* π*, outputs a bit indicating accept or reject.*
- $\mathsf{Sim}(\mathsf{crs}, \tau, \mathsf{x})$. *On input a* crs*, a trapdoor* τ *and a statement* x*, outputs a simulated proof* π *without needing a witness for* x.

Completeness. A NIZK is (perfectly) complete, if an honest prover with a valid witness can always convince an honest verifier. More formally, for any $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}_\mathcal{L}$, we have:

$$\Pr\left[\begin{array}{c}(\mathsf{crs}, \tau, \mathsf{ek}) \leftarrow \mathsf{GenCRS}(1^\lambda, \mathcal{L}), \pi \leftarrow \mathsf{P}(\mathsf{crs}, \mathsf{x}, \mathsf{w}) : \\ \mathsf{V}(\mathsf{crs}, \mathsf{x}, \pi) = 1\end{array}\right] = 1$$

Soundness. A NIZK scheme for the language $\mathcal{L}$ is called (computationally) sound, if for all PPT adversaries $\mathcal{A}$, we have:

$$\Pr\left[\begin{array}{c}(\mathsf{crs}, \tau, \mathsf{ek}) \leftarrow \mathsf{GenCRS}(1^\lambda, \mathcal{L}), (\mathsf{x}, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) : \\ \mathsf{V}(\mathsf{crs}, \mathsf{x}, \pi) = 1 \wedge \mathsf{x} \notin \mathcal{L}\end{array}\right] \approx_\lambda 0$$

Zero-knowledge. A NIZK scheme for the language $\mathcal{L}$ is called (computationally) zero-knowledge if for all PPT adversary $\mathcal{A}$,

$$\begin{aligned} &\Pr\left[\ (\mathsf{crs},\tau,\mathsf{ek}) \leftarrow \mathsf{GenCRS}(1^\lambda,\mathcal{L}) : \mathcal{A}^{\mathsf{P}(\mathsf{crs},\cdot,\cdot)}(\mathsf{crs}) = 1\ \right] \\ \approx_\lambda\ &\Pr\left[\ (\mathsf{crs},\tau,\mathsf{ek}) \leftarrow \mathsf{GenCRS}(1^\lambda,\mathcal{L}) : \mathcal{A}^{\mathsf{Sim}(\mathsf{crs},\tau,\cdot)}(\mathsf{crs}) = 1\ \right] \end{aligned}$$

Simulation Soundness. A NIZK proof for the language $\mathcal{L}$ is called simulation sound if for all PPT adversaries $\mathcal{A}$,

$$\Pr\left[\begin{array}{c} (\mathsf{crs},\tau,\mathsf{ek}) \leftarrow \mathsf{GenCRS}(1^\lambda,\mathcal{L}); (\mathsf{x},\pi) \leftarrow \mathcal{A}^{\mathsf{Sim}(\mathsf{crs},\tau,\cdot)}(\mathsf{crs}) : \\ (\mathsf{x},\pi) \notin \mathcal{Q} \wedge \mathsf{x} \notin \mathcal{L} \wedge \mathsf{V}(\mathsf{crs},\mathsf{x},\pi) = 1 \end{array}\right] \approx_\lambda 0$$

where $\mathcal{Q}$ is the list of simulation queries and responses (x_i, π_i).

Simulation extractability. This is a strong notion which requires that for any adversary that outputs a proof after seeing many simulated proofs (for either true or false statements), there exists an efficient extractor that can extract the witness from the proof. More formally, we say that a NIZK scheme for the language $\mathcal{L}$ is simulation extractable if there exists an efficient algorithm Ext (called extractor), such that for any PPT adversary $\mathcal{A}$, we have:

$$\Pr\left[\begin{array}{c} (\mathsf{crs},\tau,\mathsf{ek}) \leftarrow \mathsf{GenCRS}(1^\lambda,\mathcal{L}); (\mathsf{x},\pi) \leftarrow \mathcal{A}^{\mathsf{Sim}(\mathsf{crs},\tau,\cdot)}(\mathsf{crs},\mathsf{ek}); \\ \mathsf{w} \leftarrow \mathsf{Ext}(\mathsf{crs},\mathsf{ek},\mathsf{x},\pi) : (\mathsf{x},\pi) \notin \mathcal{Q} \wedge (\mathsf{x},\mathsf{w}) \notin \mathcal{R}_\mathcal{L} \wedge \mathsf{V}(\mathsf{crs},\mathsf{x},\pi) = 1 \end{array}\right] \approx_\lambda 0$$

where $\mathcal{Q}$ contains the list of statement-proof pairs that $\mathcal{A}$ asks the oracle $\mathsf{Sim}(\mathsf{crs},\tau,\cdot)$. Note that this definition considers a strong version of simulation extractability, where the adversary should not even be able to generate a new proof (different from oracle's response) for a statement that has been queried before. We can relax this definition by considering a weaker version where the adversary may be able to generate a new proof for a queried statements. However, for the universally composable NIZKs that we use for realizing $\mathcal{F}_{\mathsf{nizk}}$ (defined in section 3.4), we require the stronger definition. Thus, when we discuss about the notion of simulation extractability, we mean the stronger version defined above.

3.2.8 CL Framework

In [54], Castagnos and Laguillaumie introduced the so-called CL framework, which allows to construct cyclic groups $\mathbb{G}$ where the decisional Diffie-Hellman (DDH) assumption is believed to hold and furthermore, there exists a subgroup $\mathbb{H}$ of $\mathbb{G}$ such that the discrete logarithm problem in $\mathbb{H}$ is easy. As the main application of this framework, they construct the first practical linearly homomorphic encryption scheme where the plaintext space is $\mathbb{Z}_q$ with prime order q, where the size of q can be made independent of the security parameter. We make use of the CL encryption schemes in some of our constructions. In particular, the CL framework allows to encrypt values "in the exponent", so that proving statements about the message is easy using efficient Σ-protocols in [56], while at the same time allowing for efficient decryption. (As opposed to standard ElGamal encryption, where it is necessary to limit the size of the message in the exponent to allow for decryption). This will be useful both to instantiate our threshold encryption scheme (used to allow anonymity revokers to "trace" a suspicious account holder), and in our construction of extractable NIZK proofs in the crs model.

In the following, we recall the definition of CL framework from [56].

Definition

Let λ be a positive integer and q be a μ-bit prime for $\mu \geq \lambda$. The framework $\mathsf{Gen}_{\mathsf{CL}}$ consists of two algorithms $\mathsf{Gen}_{\mathsf{CL}} = (\mathsf{Gen}, \mathsf{Solve})$ defined as follows:

- $\mathsf{Gen}(1^\lambda, q) \to \mathsf{pp} := (\widehat{\mathbb{G}}, \mathbb{G}, \mathbb{H}, \mathbb{G}^q, \tilde{s}, g, h, g_q)$: the algorithm takes the security parameter λ and a prime q as inputs and outputs a tuple $\mathsf{pp} := (\widehat{\mathbb{G}}, \mathbb{G}, \mathbb{H}, \mathbb{G}^q, \tilde{s}, g, h, g_q)$, where
  - $(\widehat{\mathbb{G}}, \cdot)$ is a finite abelian group with order $\widehat{n} := q \cdot \widehat{s}$ where the bitsize of $\widehat{s}$ is a function of λ and $\gcd(q, \widehat{s}) = 1$. It is required that valid encodings of elements in $\widehat{\mathbb{G}}$ can efficiently be recognized.
  - $(\mathbb{G}, \cdot)$ is a cyclic subgroup of $\widehat{\mathbb{G}}$ of order $n := q \cdot s$ where s divides $\widehat{s}$.
  - $(\mathbb{H}, \cdot)$ is the unique cyclic subgroup of $\widehat{\mathbb{G}}$ of order q, generated by h.
  - $\mathbb{G}^q := \{x^q | x \in \mathbb{G}\}$ is the subgroup of $\mathbb{G}$ of order s. Since $\mathbb{H} \subset \mathbb{G}$, it holds that $\mathbb{G} \simeq \mathbb{G}^q \times \mathbb{H}$.
  - $\tilde{s}$ is an upper bound of $\widehat{s}$.
  - g, h and g_q are respectively the generators of $\mathbb{G}$, $\mathbb{H}$ and $\mathbb{G}^q$, where $g := h \cdot g_q$.
- $\mathsf{Solve}(q, \mathsf{pp}, X) \to x'$: this is a DPT algorithm that solves the discrete logarithm problem in $\mathbb{H}$:

$$\Pr\left[x = x' : \mathsf{pp} \leftarrow\$ \mathsf{Gen}(1^\lambda, q), x \leftarrow\$ \mathbb{Z}/q\mathbb{Z}, X \leftarrow h^x \right. \\ \left. x' \leftarrow \mathsf{Solve}\,(q, \mathsf{pp}, X)\right] = 1$$

Hard Subgroup Membership Problem

We now recall the hard subgroup membership problem within a group with an easy DL subgroup (HSM-CL) from [55]. The HSM-CL assumption states that it is hard to distinguish the elements of $\mathbb{G}^q$ in $\mathbb{G}$.

Definition 3.2.17 (HSM-CL assumption)**.** *Let λ be a positive integer and $\mathsf{Gen}_{\mathsf{CL}} = (\mathsf{Gen}, \mathsf{Solve})$ be a generator that generates a group with an easy DL subgroup as defined above. Let $\mathcal{D}$ (resp. $\mathcal{D}_q$) be a distribution over the integers such that the distance between the distribution $\{g^x, x \leftarrow\$ \mathcal{D}\}$ (resp. $\{g_q^x, x \leftarrow\$ \mathcal{D}_q\}$) and uniform distribution in $\mathbb{G}$ (resp. in $\mathbb{G}^q$) is less than $\delta(\lambda) = \mathsf{negl}(\lambda)$. The advantage of an adversary $\mathcal{A}$ for the HSM-CL problem is now defined as*

$$\mathrm{Adv}_{\mathcal{A}}^{\mathsf{hsm\text{-}cl}}(\lambda) = |\Pr[b = b' : \mathsf{pp} \leftarrow\$ \mathsf{Gen}(1^\lambda, q), |q| \geq \lambda, x \leftarrow\$ \mathcal{D}, x' \leftarrow\$ \mathcal{D}_q, \\ b \leftarrow\$ \{0,1\}, X_0 \leftarrow g^x, X_1 \leftarrow g_q^{x'}, b' \leftarrow \mathcal{A}\,(q, \mathsf{pp}, X_b, \mathsf{Solve}(\cdot))] - 1/2|$$

We say that the HSM-CL problem is ε-hard for $\mathsf{Gen}_{\mathsf{CL}}$ if for all PPT adversary $\mathcal{A}$, $\mathrm{Adv}_{\mathcal{A}}^{\mathsf{hsm\text{-}cl}}(\lambda) \leq \varepsilon(\lambda)$. And we say HSM-CL holds for $\mathsf{Gen}_{\mathsf{CL}}$ if it is ε-hard for $\mathsf{Gen}_{\mathsf{CL}}$ and $\varepsilon(\lambda) = \mathsf{negl}(\lambda)$.

PKE scheme under HSM-CL assumption

We recall the scheme described in [55] with the following setting: the plaintext space is $\mathbb{Z}_q$ for μ-bit prime q and $\mu \geq \lambda$; and the secret key x and randomness r are drawn from distribution $\mathcal{D}_q$. As shown in [55] (lemma 4), to instantiate $\mathcal{D}_q$ in practice, one can use the uniform distribution on $\{0, \ldots, S\}$ for $S := 2^{\lambda-2} \cdot \tilde{s}$. The scheme is depicted in Fig.

$\mathsf{CL.Setup}(1^\lambda, 1^\mu)$

1. sample a μ-bit prime q
2. $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda, q)$
3. return pp

$\mathsf{CL.KeyGen}(\mathsf{pp})$

1. sample $\alpha \leftarrow\!\!\$\ \mathcal{D}_q$ and $y \leftarrow g_q^\alpha$
2. set $\mathsf{pk} := y$ and $\mathsf{sk} := \alpha$
3. return $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{CL.Enc}(\mathsf{pk}, m)$

1. sample $r \leftarrow\!\!\$\ \mathcal{D}_q$
2. return $(g_q^r, h^m y^r)$

$\mathsf{CL.Dec}(\mathsf{sk}, (c_1, c_2))$

1. compute $M \leftarrow c_2 / c_1^\alpha$
2. return $\mathsf{Solve}(q, \mathsf{pp}, M)$

Figure 3.6: Encryption scheme from HSM-CL [55]

Theorem 3.2.18. *[55] The scheme described in Fig. 3.6 is semantically secure under chosen plaintext attacks (IND-CPA) under the HSM-CL assumption.*

3.3 System Design

We give a high-level overview of the design of the identity layer in terms of the entities involved, data objects and protocols between the entities.

3.3.1 Entities Involved

The following entities are involved in our design:

- Account Holders (AH): those are individuals who hold accounts on the block-chain. We assume AHs possess some mean for performing legal identification (e.g., a passport), in the country where they live. They are interested in opening accounts and performing transactions on the blockchain but, before doing so, they have to register with an Identity Provider (IP).
- Identity Provider (IP): an identity provider is an entity that, as the name suggests, can provide a digital identity to an AH. The identity provider "authorizes" a user to open

accounts on the blockchain, and therefore to perform transactions. Jumping ahead, when observing transactions on the blockchain, it should not be possible to find out the identity of an AH (not even for the IP itself), while everyone should be able to see which IP has authorized a given account, thus creating trust in the account.

- Anonymity Revoker (AR): anonymity revokers are parties which are involved in case where law-enforcement or other authorized entities need to be able to extract the identity of the owner of some account on the blockchain. We can make threshold assumptions on the AR and e.g., require that at least $\mathsf{d}+1$ ARs must give an approval before the anonymity of a user is revoked.

3.3.2 Data Objects

We now describe the data objects that are held by the entities.

*Account Holder Certificate (*AHC*).* After an account holder registers with an identity provider, the AH obtains a certificate containing:

- A public identity credential $\mathsf{IDcred}_{\mathtt{PUB}}$ and a secret identity credential $\mathsf{IDcred}_{\mathtt{SEC}}$.
- A key K for a pseudorandom function PRF.
- One or more attribute lists AL such as some identifier, age, citizenship, expiration date, etc.
- A signature on $(\mathsf{IDcred}_{\mathtt{SEC}}, \mathsf{K}, \mathsf{AL})$ that can be checked using $\mathsf{pk}_{\mathsf{IP}}$. A valid signature proves that an AH with attributes as in AL has registered with IP and has proved knowledge of $\mathsf{IDcred}_{\mathtt{SEC}}$ corresponding to $\mathsf{IDcred}_{\mathtt{PUB}}$.

*Account Creation Information (*ACI*).* Given an AHC, an account holder can create new accounts and post the corresponding ACI on the ledger, containing:

- $\mathsf{RegID}_{\mathsf{ACC}}$, an account registration ID. This is defined to be $\mathsf{RegID}_{\mathsf{ACC}} = \mathsf{PRF}_{\mathsf{K}}(x)$ where K is a key held by AH and signed by the IP, and where the account in question is the x'th account opened by the AH based on a given AHC. If AH behaves honestly, then $\mathsf{RegID}_{\mathsf{ACC}}$ is unique for the account, and $x \leq \mathsf{Max}_{\mathsf{ACC}}$. The latter condition is enforced by the proof below, the former can be checked publicly.
- Anonymity revocation data: this is a threshold encryption $\mathsf{E}_{\mathsf{ID}} = \mathsf{TEnc}^{\mathsf{n},\mathsf{d}}_{\mathsf{PK}_{\mathsf{AR}}}(\mathsf{IDcred}_{\mathtt{PUB}})$, where any subset of size $\mathsf{d}+1$ of anonymity revokers are able to decrypt E_{ID} and obtain $\mathsf{IDcred}_{\mathtt{PUB}}$.
- The identity IP of the identity provider who did the signature in the AHC used for this account.
- An account specific public key $\mathsf{pk}_{\mathsf{ACC}}$. It will be used, for instance, to verify transactions related to the account.
- A policy P, which asserts some information about the attribute list AL.

- A proof π that can be checked using $\mathsf{pk}_{\mathsf{IP}}$ and verifies that ACI can only be created by an AH that has obtained an AHC from IP, such that $\mathsf{P}(\mathsf{AL}) = \top$, where AH knows the secret keys corresponding to $\mathsf{pk}_{\mathsf{ACC}}$, as well as $\mathsf{IDcred}_{\mathtt{SEC}}$ corresponding to the $\mathsf{IDcred}_{\mathtt{PUB}}$ that was presented to the IP, and where $\mathsf{RegID}_{\mathsf{ACC}}, \mathsf{E}_{\mathsf{ID}}$ and $\mathsf{E}_{\mathsf{RegID}} = \mathsf{TEnc}^{\mathsf{n,d}}_{\mathsf{PK}_{\mathsf{AR}}}(\mathsf{K})$ are correctly generated.

*Identity Provider's information on Account Holder (*IPIAH*).* This is the data record that the IP stores after an AH has registered. It contains:

- The name AH of the account holder and its public identity credential $\mathsf{IDcred}_{\mathtt{PUB}}$.
- A set of anonymity revokers $\mathsf{AR}_1, ..., \mathsf{AR}_\mathsf{n}$ with public keys $\mathsf{PK}_{\mathsf{AR}}$ and an encryption $\mathsf{E}_{\mathsf{RegID}} = \mathsf{TEnc}^{\mathsf{n,d}}_{\mathsf{PK}_{\mathsf{AR}}}(\mathsf{K})$. Here, K is the PRF key chosen by the AH at registration time.

3.3.3 Protocols

The following are the main protocols in our design.

Account Holder Registration. The protocol takes place between an IP and an AH who owns a key pair $(\mathsf{IDcred}_{\mathtt{SEC}}, \mathsf{IDcred}_{\mathtt{PUB}})$ and an attribute list AL. At the end of the protocol, the AH receives an AHC and the IP obtains a IPIAH as described above. The AH sends their attribute list AL to IP and proves (via non-cryptographic means) their identity to IP. More concretely, this means that the IP must verify that the entity it is talking to indeed has the name AH and hence it received AL from the correct entity. It should also verify that the attributes in AL are correct w.r.t. the AH. The AH also sends to IP their public key $\mathsf{IDcred}_{\mathtt{PUB}}$ and an encryption $\mathsf{E}_{\mathsf{RegID}} = \mathsf{TEnc}^{\mathsf{n,d}}_{\mathsf{PK}_{\mathsf{AR}}}(\mathsf{K})$ where K is a PRF key. Next, AH and IP engage in a blind signature scheme, which allows AH to receive a signature on $(\mathsf{IDcred}_{\mathtt{SEC}}, \mathsf{K}, \mathsf{AL})$ that is generated under the secret key $\mathsf{sk}_{\mathsf{IP}}$ of the IP. In addition, AH proves (cryptographically, in ZK) that they know $\mathsf{IDcred}_{\mathtt{SEC}}$ corresponding to $\mathsf{IDcred}_{\mathtt{PUB}}$, that the same $\mathsf{IDcred}_{\mathtt{SEC}}$ was input to the blind signature, and that the encryption contains the same K that was input to the blind signature scheme. IP stores $\mathsf{IPIAH} = (\mathsf{ID}_{\mathsf{AH}}, \mathsf{IDcred}_{\mathtt{PUB}}, \mathsf{AL}, \mathsf{E}_{\mathsf{RegID}}, \mathsf{AR}_1, \ldots, \mathsf{AR}_\mathsf{n})$.
Create New Account. An account holder AH wants to create an account that satisfies some policy P (e.g., above 18, resident in country X, etc.). They take as input an AHC, a policy P and the public key $\mathsf{pk}_{\mathsf{AR}}$ of one (or more) anonymity revoker(s) with name AR. At the end, AH produces some ACI that can be posted to the blockchain. They also need to store secret key $\mathsf{sk}_{\mathsf{ACC}}$ that is specific to the account. The protocol works as follows: AH generates an account key pair $(\mathsf{pk}_{\mathsf{ACC}}, \mathsf{sk}_{\mathsf{ACC}})$ and an encryption of their public identity credential $\mathsf{IDcred}_{\mathtt{PUB}}$ under the public key of the anonymity revokers' $\mathsf{PK}_{\mathsf{AR}}$, i.e., $\mathsf{E}_{\mathsf{ID}} = \mathsf{TEnc}^{\mathsf{n,d}}_{\mathsf{PK}_{\mathsf{AR}}}(\mathsf{IDcred}_{\mathtt{PUB}})$. Next, AH calculates $\mathsf{RegID}_{\mathsf{ACC}} = \mathsf{PRF}_{\mathsf{K}}(x)$, where we assume this is the x'th account that is opened using the AHC that is input. At last, AH produces a a non-interactive zero-knowledge (NIZK) proof of knowledge π for statement

$$st = (\mathsf{P}, \mathsf{E}_{\mathsf{ID}}, \mathsf{RegID}_{\mathsf{ACC}}, \mathsf{IP}, \mathsf{pk}_{\mathsf{ACC}})$$

using secret witness

$$w = (\sigma, \mathsf{IDcred}_{\mathtt{SEC}}, \mathsf{K}, \mathsf{AL}, \mathsf{sk}_{\mathsf{ACC}})$$

for the relation $R(st, w)$ that outputs $\top$ if:

1. σ is a valid signature under $\mathsf{pk}_{\mathsf{IP}}$ for a message of form $(\mathsf{IDcred}_{\mathtt{SEC}}, \mathsf{K}, \mathsf{AL})$.
2. AL satisfies the policy i.e., $\mathsf{P}(\mathsf{AL}) = \top$.
3. $\mathsf{RegID}_{\mathsf{ACC}} = \mathsf{PRF}_{\mathsf{K}}(x)$ for some $x \leq \mathsf{Max}_{\mathsf{ACC}}$.
4. $\mathsf{E}_{\mathsf{ID}} = \mathsf{TEnc}^{\mathsf{n},\mathsf{d}}_{\mathsf{PK}_{\mathsf{AR}}}(\mathsf{IDcred}_{\mathtt{PUB}})$.
5. $(\mathsf{pk}_{\mathsf{ACC}}, \mathsf{sk}_{\mathsf{ACC}})$ is a valid key pair.

Let $\mathsf{ACI} = (\mathsf{RegID}_{\mathsf{ACC}}, \mathsf{E}_{\mathsf{ID}}, \mathsf{AR}_1, ..., \mathsf{AR}_{\mathsf{n}}, \mathsf{IP}, \mathsf{pk}_{\mathsf{ACC}}, \mathsf{P}, \pi)$.
Revoke Anonymity of Account. Revocation of the anonymity of an account can be done by at least $\mathsf{d}+1$ of the n ARs involved in the set-up of the account, working together with the IP with whom the AH registered. The input is an account identifier $\mathsf{RegID}_{\mathsf{ACC}}$ and the output is the name AH of the account holder. The protocol proceeds as follows: Given an account $\mathsf{RegID}_{\mathsf{ACC}}$ whose anonymity needs to be revoked, the ARs find the ACI containing $\mathsf{RegID}_{\mathsf{ACC}}$ on the blockchain, collaborate to decrypt E_{ID} and learn $\mathsf{IDcred}_{\mathtt{PUB}}$. The registration information also contains the public name IP of the identity provider who registered $\mathsf{IDcred}_{\mathtt{PUB}}$. The AR's contact this IP who then locates the $\mathsf{IPIAH} = (\mathsf{AH}, \mathsf{IDcred}_{\mathtt{PUB}}, \mathsf{AL}, \mathsf{E}_{\mathsf{RegID}}, \mathsf{AR}_1, ..., \mathsf{AR}_{\mathsf{n}})$ record that contains the $\mathsf{IDcred}_{\mathtt{PUB}}$ that was decrypted. This record also includes AH, thus IP and the set of ARs have now identified the AH.
Trace accounts of User. If a user with a given name AH is suspected of engaging in illegal activities, the IP and a set of at least $\mathsf{d}+1$ ARs can identify all accounts of that user. The IP searches its database to locate the $\mathsf{IPIAH} = (\mathsf{AH}, \mathsf{IDcred}_{\mathtt{PUB}}, \mathsf{AL}, \mathsf{E}_{\mathsf{RegID}}, \mathsf{AR}_1, ..., \mathsf{AR}_{\mathsf{n}})$ containing the relevant AH. This record also contains the names of the relevant AR's. A qualified set of these could decrypt the $\mathsf{E}_{\mathsf{RegID}}$ to learn the PRF key K and generate all values $\mathsf{PRF}_{\mathsf{K}}(x)$ for $x = 1, \ldots, \mathsf{Max}_{\mathsf{ACC}}$ in public. However, due to technicalities in the security reduction, this would require the PRF to satisfy some form of "selective opening attack" security. Instead, we let the AR's decrypt the ciphertext and evaluate the PRF on $x = 1, \ldots, \mathsf{Max}_{\mathsf{ACC}}$ inside an MPC protocol, so that K is never revealed to anyone. Either way, the produced values are all the possible values for $\mathsf{RegID}_{\mathsf{ACC}}$ that the AH could have used to form valid accounts, so one can now search the blockchain for accounts with these registration IDs.

3.3.4 Informal Analysis of the Design

If an AH misbehaves and opens more accounts than they are allowed to, this must result in two or more accounts with the same $\mathsf{RegID}_{\mathsf{ACC}}$. This can be publicly detected by the blockchain, and the second account will be discarded. We note that for this to work, we assume that incentives have been created so that some parties will indeed observe the duplicates and alert the relevant entities. Moreover, the construction satisfies *revocability* and *traceability*, meaning a malicious AH cannot create a valid account such that the anonymity revokers together with the identity provider are unable to revoke its anonymity or trace it. This follows from the soundness of the underlying zero-knowledge proofs which imply $\mathsf{E}_{\mathsf{ID}} = \mathsf{TEnc}^{\mathsf{n},\mathsf{d}}_{\mathsf{PK}_{\mathsf{AR}}}(\mathsf{IDcred}_{\mathtt{PUB}})$ and $\mathsf{E}_{\mathsf{RegID}} = \mathsf{TEnc}^{\mathsf{n},\mathsf{d}}_{\mathsf{PK}_{\mathsf{AR}}}(\mathsf{K})$. Thus, any subset of size $\mathsf{d}+1$ of anonymity revokers can decrypt E_{ID} (resp. $\mathsf{E}_{\mathsf{RegID}}$) and revoke the AH's anonymity (resp. trace all the AH's accounts). Lastly, due to the security of the underlying PRF and the threshold encryption scheme and also the ZK property of the proof $\pi \in \mathsf{ACI}$, our design supports *anonymity* of the account holders, in the sense that a malicious identity provider even by cooperating with d anonymity revokers and other dishonest account holders cannot link

a valid account to an account holder. Since we are using a Blockchain e.g., an imperfect bulletin board, we also need to worry that a malicious AH can't "rush" and steal an honest user account number by maliciously choosing a PRF key K which "hits" some of the account numbers of the honest users which have not yet been finalized by the Blockchain. In order to do this we define and use a *weakly robust* PRF.

3.4 ID-layer Formalization

3.4.1 The UC-Model

We use the UC-security [52] framework with static corruption. In the following, the reader is assumed to be familiar with the basic concepts of UC security and is referred to [52] for a more detailed description.

Let $\mathcal{Z}$ denote the environment. For a protocol Π and an adversary $\mathcal{A}$, we write $\mathsf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}$ to denote the ensemble corresponding to the protocol execution. For an ideal functionality $\mathcal{F}$ and a simulator $\mathcal{S}$, we write $\mathsf{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ to denote the distribution ensemble of the ideal world execution. We say that a protocol Π UC-realizes a functionality $\mathcal{F}$ if for all PPT adversaries $\mathcal{A}$ corrupting a subset of parties, there exists a simulator $\mathcal{S}$ such that for all environments $\mathcal{Z}$, the ensembles $\mathsf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}$ and $\mathsf{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ are computationally indistinguishable.

In the rest of the section we first recall (standard) ideal functionalities used by our protocol and then describe the main ideal functionalities in our construction.

Functionality $\mathcal{F}_{\mathsf{crs}}$

The functionality is parametrized by a distribution $\mathcal{D}$ and proceeds as follows.

- Choose a value $\mathsf{crs} \leftarrow\!\!\$\ \mathcal{D}$.
- On input $(\mathtt{CRS})$ from a party P, return $(\mathtt{CRS}, \mathsf{crs})$ to P.

Functionality $\mathcal{F}_{\mathsf{smt}}$

The functionality models a secure channel between a sender S and a receiver R.

- Upon input $(\mathtt{SEND}, R, m)$ from a party S, if both S and R are honest, output $(\mathtt{SENT}, S, m)$ to R and $(\mathtt{SENT}, S, R, |m|)$ to $\mathcal{A}$. Otherwise, if at least one of S and R is corrupt, output $(\mathtt{SENT}, S, R, m)$ to $\mathcal{A}$.

Registration Functionality

The functionality allows identity providers and anonymity revokers to input a key pair $(\mathsf{sk}, \mathsf{pk}) \in \{0,1\}^* \times \{0,1\}^*$ once such that they would not be allowed to modify or delete it later. The account holders can retrieve the public keys of registered parties by the $\mathtt{RETRIEVE}$ command.

Functionality $\mathcal{F}_{\mathsf{reg}}$

Register

Upon receiving a message $(\texttt{REGISTER}, \mathsf{sk}_P, \mathsf{pk}_P)$ from party P, if this is the first request from P, then keep the record $(P, \mathsf{sk}_P, \mathsf{pk}_P)$ and return $(\texttt{INITIALIZED}, P)$. Otherwise, ignore the message.

Retrieve

Upon receiving a message $(\texttt{RETRIEVE}, P_i)$ from some party P_j or the adversary, output $(\texttt{RETRIEVE}, P_i, \mathsf{pk}_{P_i})$ to P_j where $\mathsf{pk}_{P_i} = \perp$ if no record $(P_i, \mathsf{sk}_{P_i}, \mathsf{pk}_{P_i})$ exists.

NIZK Functionality

We use $\mathcal{F}_{\mathsf{nizk}}$ as defined by Groth et al. in [119].

Functionality $\mathcal{F}_{\mathsf{nizk}}$

The functionality is parametrized by a relation $\mathcal{R}$ for which we can efficiently check membership.

Proof

- On input $(\texttt{PROVE}, x, w)$ from party P, ignore if $(x, w) \notin \mathcal{R}$. Send $(\texttt{PROVE}, x)$ to $\mathcal{A}$.
- Upon receiving the answer $(\texttt{PROOF}, \pi)$ from $\mathcal{A}$, store (x, π) and send $(\texttt{PROOF}, \pi)$ to P.

Verify

- On input $(\texttt{VERIFY}, x, \pi)$ from V check whether (x, π) is stored. If not send $(\texttt{VERIFY}, x, \pi)$ to $\mathcal{A}$.
- Upon receiving the answer $(\texttt{WITNESS}, w)$ from $\mathcal{A}$, check $(x, w) \in \mathcal{R}$ and if so, store (x, π). If (x, π) has been stored, output $(\texttt{VERIFICATION}, 1)$ to V, else output $(\texttt{VERIFICATION}, 0)$.

Ledger Functionality

The functionality $\mathcal{F}_{\mathsf{ledger}}$ (which is an adapted and simplified version of the functionality given by [126]) abstracts a public ledger which can be accessed globally by protocol parties or environment $\mathcal{Z}$. When accounts are created and posted by account holders, the ideal functionality first validates the account and then appends them to the buffer. The environment can later specify when to free the buffer and append it to the ledger.

Functionality $\mathcal{F}_{\mathsf{ledger}}$

The functionality is globally available to all parties and is parameterized by a predicate VALIDATE, an initially empty list L of bit strings and a variable buffer initially set to ε.

Append

Upon input $(\texttt{APPEND}, x)$ from party P, if $\texttt{VALIDATE}(\mathsf{state}, (\mathsf{buffer}, x)) = 1$, then set $\mathsf{buffer} \leftarrow \mathsf{buffer}||x$.

Retrieve

Upon input $(\texttt{RETRIEVE})$ from a party P or $\mathcal{A}$, return $(\texttt{RETRIEVE}, L)$ to the requestor in case of an honest party or $(\texttt{RETRIEVE}, L, \mathsf{buffer})$ in case of a corrupt party.

Buffer Release

Upon input $(\texttt{RELEASE}, \Pi)$ from $\mathcal{A}$, append a permutation Π of buffer to L by setting $L \leftarrow L||\Pi(\mathsf{buffer})$ and set $\mathsf{buffer} := \varepsilon$.

Functionality for MPC of PRF output

This functionality allows a set of parties who own secret shares of a PRF key to evaluate the PRF on a fixed set of inputs.

Functionality $\mathcal{F}_{\mathsf{mpc\text{-}prf}}$

The functionality interacts with parties $P_1, \ldots, P_{\mathsf{n}}$ and is parameterized by a pseudo-random function PRF, a constant $\mathsf{Max}_{\mathsf{ACC}}$, and a (n, d)-secret sharing scheme **SS**.

Compute

Upon input $(\texttt{COMPUTE}, \mathsf{K}_i)$ from at least $\mathsf{d}+1$ distinct P_i, proceed as follows:

- compute $\mathsf{K} = \mathsf{Reconstruct}^{\mathsf{n},\mathsf{d}}(\mathsf{K}_{i_1}, \ldots, \mathsf{K}_{i_{\mathsf{d}+1}})$.
- evaluate $\mathsf{PRF}_{\mathsf{K}}(x)$ on all inputs $x \in [\mathsf{Max}_{\mathsf{ACC}}]$ and return the list of outputs to all the requesters P_i for $i \in \mathsf{n}$.

3.4.2 The ID Layer Functionality

The functionality $\mathcal{F}_{\mathsf{id\text{-}layer}}$ captures the security properties offered by the design of our identity layer while hiding the implementation details. After the functionality is initialized, it allows identity providers IP to issue credentials to account holders AH based on their attribute lists AL. At the level of the ideal functionality, a credential is just a pointer to a record storing the tuple $(\mathsf{AH}, \mathsf{IP}, \mathsf{AL})$. Armed with a credential, an AH can create up to $\mathsf{Max}_{\mathsf{ACC}}$ accounts. When creating an account, the AH can choose a predicate P of their attributes to be made public (e.g., "I am over 18, I am resident of country X" etc.) which, together with the IP who authorized this account, are the only information which are made public. We capture this by having the

functionality leak only the fact that an account was created and not the identity of the AH.[5] Moreover, when creating an account, the AH also registers a key-pair associated to this account. The functionality is parametrized by any key-pair relation, which allows our ideal functionality to be used as a building block in more complex protocols, where the AH then can use those keys for authentication, encryption, etc. Our functionality also exposes some of the details about the underlying ledger on top of which it is implemented, thus new accounts are added to a buffer which can be permuted by the adversary before becoming finalized. This is inevitable as we run this on top of a ledger which has the same properties. The final two commands of the ideal functionality, revoke and trace, allow a qualified set of anonymity revokers AR and an IP to respectively disclose the AH behind a given account, or to find all accounts belonging to a certain AH.

Functionality Identity layer $\mathcal{F}_{\mathsf{id\text{-}layer}}$

We assume that $\{\mathsf{IP}_1, \ldots, \mathsf{IP}_\mathsf{m}, \mathsf{AR}_1, \ldots, \mathsf{AR}_\mathsf{n}\}$ is the set of identifiers for identity providers and anonymity revokers. The functionality is parameterized by values m, n and threshold d, together with an NP (key-pair) relation $\mathcal{R}_{\mathsf{ACC}}$ such that when parties input `CreateACC`, they also specify a key-pair and the functionality verifies if the key-pair satisfies $\mathcal{R}_{\mathsf{ACC}}$. Moreover, the functionality maintains the following initially empty records: Count, where $\mathsf{Count}[\mathsf{cid}]$ counts the number of accounts created by certificate cid, and two records Cert and ACC, respectively for keeping track of certificates and accounts and a list L of public account information.

Initialize

On $(\texttt{INITIALIZE})$ from party $P \in \{\mathsf{IP}_1, \ldots, \mathsf{IP}_\mathsf{m}, \mathsf{AR}_1, \ldots, \mathsf{AR}_\mathsf{n}\}$, output to $\mathcal{A}$ $(\texttt{INITIALIZED}, P)$.

If all parties have been initialized, store $(\texttt{READY})$.

Issue

On $(\texttt{ISSUE}, \mathsf{IP}, \mathsf{AL})$ from an honest account holder AH (or the adversary in the name of corrupted account holder AH) and input $(\texttt{ISSUE}, \mathsf{AH})$ from identity provider IP:

- If not $(\texttt{READY})$, then ignore.
- If there is already a cid with $\mathsf{Cert}[\mathsf{cid}] = (\mathsf{AH}, \mathsf{IP}, \cdot, \cdot)$, then abort; otherwise, if IP is honest (resp. corrupt), then send $(\texttt{ISSUE})$ (resp. $(\texttt{ISSUE}, \mathsf{AH}, \mathsf{IP}, \mathsf{AL})$) to $\mathcal{A}$.

[5] Note that the environment provides all inputs and sees all outputs. It can therefore observe that an account is created right after it instructed an account holder to create an account, and can make the connection between the two. This corresponds to the fact that in a real application an adversary may know that in a long time interval, only one user creates an account, and so the next account that shows up on chain must belong to that user. Of course, our system cannot prevent this - the best we can do is to make sure that the account itself is anonymous. This follows in our model because the ideal adversary - the simulator - will not learn the identity of the holder and will still have to produce account information which are indistinguishable from the real protocol, thus proving that the account information leaks no information about its holder.

- Upon receiving $(\mathsf{cid}, \texttt{ISSUE})$ from $\mathcal{A}$, if $\mathsf{cid} = \perp$ (in the case of corrupt IP) or if there already exists cid s.t. $\mathsf{Cert}[\mathsf{cid}] \neq \perp$, then abort. Otherwise, set $\mathsf{Cert}[\mathsf{cid}] \leftarrow (\mathsf{AH}, \mathsf{IP}, \mathsf{AL})$ and output $(\texttt{ISSUED}, \mathsf{cid})$ to AH.

Account Creation

Upon inputs $(\texttt{CreateACC}, \mathsf{cid}, \mathsf{P}, (\mathsf{sk}_{\mathsf{ACC}}, \mathsf{pk}_{\mathsf{ACC}}))$ from honest account holders AH (or the adversary in the name of corrupted account holder AH), if not $(\texttt{READY})$, then ignore. Else, proceed as follows:

- If $\mathsf{Cert}[\mathsf{cid}] = \perp$ then abort, else retrieve $(\mathsf{AH}', \mathsf{IP}, \mathsf{AL}) \leftarrow \mathsf{Cert}[\mathsf{cid}]$.
- Check if $\mathsf{AH}' = \mathsf{AH}$ and $\mathsf{Count}[\mathsf{cid}] < \mathsf{Max}_{\mathsf{ACC}}$ and that AL satisfies the policy P.
- Verify that the key pair $(\mathsf{sk}_{\mathsf{ACC}}, \mathsf{pk}_{\mathsf{ACC}})$ satisfies the relation $\mathcal{R}_{\mathsf{ACC}}$ and abort otherwise.
- Output $(\texttt{CreateACC}, \mathsf{P}, \mathsf{pk}_{\mathsf{ACC}}, \mathsf{IP})$ to $\mathcal{A}$.
- Receiving a response $(\texttt{CreateACC}, \mathsf{aid})$ from $\mathcal{A}$, if $\mathsf{aid} = \perp$ or $\mathsf{ACC}[\mathsf{aid}] \neq \perp$, then abort, else do the followings:
  - set $\mathsf{ACC}[\mathsf{aid}] \leftarrow (\mathsf{cid}, \mathsf{P}, \mathsf{sk}_{\mathsf{ACC}})$.
  - set $\mathsf{Count}[\mathsf{cid}] \leftarrow \mathsf{Count}[\mathsf{cid}] + 1$.
  - add the tuple $(\mathsf{aid}, \mathsf{P}, \mathsf{pk}_{\mathsf{ACC}}, \mathsf{IP})$ to the account buffer.
- Return $(\texttt{Created}, \mathsf{aid})$ to AH.

Account Buffer Release

Upon input $(\texttt{RELEASE}, \Pi)$ from the adversary $\mathcal{A}$, remove all tuples from the account buffer and add the permuted tuples $(\mathsf{aid}, \mathsf{P}, \mathsf{pk}_{\mathsf{ACC}}, \mathsf{IP})$ of accounts to the account list L.

Accounts Retrieve

On $(\texttt{RETRIEVE})$ from an account holder or party $P \in \{\mathsf{IP}_1, \ldots, \mathsf{IP}_\mathsf{m}, \mathsf{AR}_1, \ldots, \mathsf{AR}_\mathsf{n}\}$, output a list including all existing tuples $(\mathsf{aid}, \mathsf{P}, \mathsf{pk}_{\mathsf{ACC}}, \mathsf{IP})$ in the account list L.

Revoke

Upon input $(\texttt{REVOKE}, \mathsf{aid})$ from a (possibly corrupt) identity provider IP and a set of (possibly corrupt) anonymity revokers $\{\mathsf{AR}_i\}_{i \in I \subseteq [\mathsf{n}]}$, proceed as follows:

- If $\mathsf{ACC}[\mathsf{aid}] = \perp$ then return $\perp$. Otherwise, retrieve $(\mathsf{cid}, \mathsf{P}, \mathsf{sk}_{\mathsf{ACC}}) \leftarrow \mathsf{ACC}[\mathsf{aid}]$.
- If IP is the same as the identity provider in $\mathsf{Cert}[\mathsf{cid}]$ and $|I| > \mathsf{d}$, then return $(\mathsf{aid}, \mathsf{AH})$ to the IP and $\{\mathsf{AR}_i\}_{i \in I}$. Otherwise, return $\perp$. Moreover, if the identity provider or at least one anonymity revoker is corrupt, output $(\mathsf{aid}, \mathsf{AH})$ to $\mathcal{A}$ as well.

Trace

Upon input $(\texttt{TRACE}, \mathsf{AH})$ from a (possibly corrupt) identity provider IP and a set of (possibly corrupt) anonymity revokers $\{\mathsf{AR}_i\}_{i \in I}$, proceed as follows:

- If there is no record $(\mathsf{AH}, \mathsf{IP}, \cdot, \cdot)$ in Cert, then return $\perp$. Otherwise, retrieve $(\mathsf{AH}, \mathsf{IP}, \cdot) \leftarrow \mathsf{Cert}[\mathsf{cid}]$.
- If $|I| > \mathsf{d}$, return to IP and $\{\mathsf{AR}_i\}_{i \in I}$ the list of all aid's such that $\mathsf{ACC}[\mathsf{aid}] = (\mathsf{cid}, \cdot, \cdot)$. Moreover, if the identity provider or at least one anonymity revoker is corrupt, return the list to $\mathcal{A}$ as well.

3.4.3 Issuing Credentials – the Functionality

We describe here $\mathcal{F}_{\mathsf{issue}}$, an ideal functionality capturing the desired properties of the issue protocol, which allows an identity provider to issue credentials to account holders. Note that the functionality can be seen as an augmented blind signature functionality: the account holder receives a signature (under the secret key of the identity provider) on a secret message m (as in blind signatures) but also on some public auxiliary information aux and on a secret key chosen by the account holder. The identity provider is not supposed to learn m (as in blind signatures), but in addition the identity provider learns a ciphertext which is guaranteed to contain an encryption of m and the public key corresponding to the secret key which is being signed.

Functionality Issue $\mathcal{F}_{\mathsf{issue}}^{\mathcal{R},\mathbf{TE},\mathbf{SIG}}$

The functionality is parametrized by an NP relation $\mathcal{R}$ corresponding to the account holders key pair, a signature scheme $\mathbf{SIG} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{VerifySig})$ and a (n, d)-threshold encryption scheme $\mathbf{TE} = (\mathsf{TKeyGen}, \mathsf{TEnc}, \mathsf{TDec})$. We assume that $\{\mathsf{IP}_1, \ldots, \mathsf{IP}_\mathsf{m}\}$ is the set of identifiers for identity providers.

Setup

- Upon input $(\texttt{SETUP})$ from any party, and only once, run $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ and return $(\texttt{SETUPREADY}, \mathsf{pp})$ to all parties.

Initialize

- Upon input $(\texttt{INITIALIZE}, (\mathsf{sk}_\mathsf{IP}, \mathsf{pk}_\mathsf{IP}))$ from identity provider IP, ignore if the party is already initialized or if $\texttt{SETUPREADY}$ has not been returned yet. Otherwise, if $(\mathsf{sk}_\mathsf{IP}, \mathsf{pk}_\mathsf{IP})$ is a valid key pair according to the relation defined by $\mathsf{KeyGen}(\mathsf{pp})$, store $(\mathsf{sk}_\mathsf{IP}, \mathsf{pk}_\mathsf{IP})$ for this party and output $(\texttt{INITIALIZED}, \mathsf{pk}_\mathsf{IP}, \mathsf{IP})$ to $\mathcal{A}$.
- If all parties have been initialized, store $(\texttt{READY})$.

Issue

On input $\Big(\texttt{ISSUE}, (ct, m, r, \mathsf{pk}_\mathsf{AR}), \mathsf{aux}, (\mathsf{sk}_\mathsf{AH}, \mathsf{pk}_\mathsf{AH}), \mathsf{IP}\Big)$ from account holder AH and input $(\texttt{ISSUE}, \mathsf{AH}, \mathsf{pk}_\mathsf{AR})$ from identity provider IP:

- If not $(\texttt{READY})$, then ignore.
- If $(\mathsf{sk}_\mathsf{AH}, \mathsf{pk}_\mathsf{AH}) \notin \mathcal{R}$ or $ct \neq \mathsf{TEnc}^{\mathsf{n},\mathsf{d}}_{\mathsf{PK}_\mathsf{AR}}(m; r)$ then abort.

- Otherwise compute $\sigma \leftarrow\!\!\$\ \mathsf{Sign}((\mathsf{sk}_{\mathsf{AH}}, m, \mathsf{aux}), \mathsf{sk}_{\mathsf{IP}})$.
- Output σ to AH and $(\mathsf{pk}_{\mathsf{AH}}, \mathsf{aux}, ct)$ to IP.

3.5 Formal Protocols Specifications

3.5.1 Identity Layer Protocol

The protocol $\Pi_{\mathsf{id\text{-}layer}}$ is run by parties interacting with ideal functionalities $\mathcal{F}_{\mathsf{reg}}$, $\mathcal{F}_{\mathsf{issue}}$, $\mathcal{F}_{\mathsf{nizk}}$, $\mathcal{F}_{\mathsf{crs}}$, $\mathcal{F}_{\mathsf{ledger}}$ and $\mathcal{F}_{\mathsf{mpc\text{-}prf}}$. Let $\mathcal{R}$ and $\mathcal{R}_{\mathsf{ACC}}$ be NP relations corresponding to the account holders' key-pair and accounts' key-pair, respectively. Let $\mathbf{TE} = (\mathsf{TKeyGen}, \mathsf{TEnc}, \mathsf{TDec})$ denote a threshold encryption scheme, PRF a pseudorandom function and $\mathbf{SIG} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{VerifySig})$ a signature scheme. Protocol $\Pi_{\mathsf{id\text{-}layer}}$ proceeds as follows.

Protocol Identity layer $\Pi_{\mathsf{id\text{-}layer}}$

Parameters for ideal functionalities.

- We use a $\mathcal{F}_{\mathsf{ledger}}$ that implements the following `VALIDATE` predicate: the predicate accepts if the NIZK proof π is valid and if $\mathsf{RegID}_{\mathsf{ACC}}$ has not been seen before.
- We use a $\mathcal{F}_{\mathsf{crs}}$ functionality that outputs the public parameters for the signature scheme and the threshold encryption scheme.

The protocol description for an account holder AH**.**

- On input $(\mathtt{ISSUE}, \mathsf{IP}, \mathsf{AL})$, retrieve the public key $\mathsf{PK}_{\mathsf{IP}}$ of identity provider IP and the vector $\mathsf{PK}_{\mathsf{AR}}$ of all public keys of the anonymity revokers via $\mathcal{F}_{\mathsf{reg}}$ and proceed as follows:
  - Generate a key pair $(\mathsf{IDcred}_{\mathtt{SEC}}, \mathsf{IDcred}_{\mathtt{PUB}})$ satisfying $\mathcal{R}$.
  - Choose a random key K for PRF and compute the encryption $\mathsf{E}_{\mathsf{RegID}} = \mathsf{TEnc}^{\mathsf{n,d}}_{\mathsf{PK}_{\mathsf{AR}}}(\mathsf{K}; r)$ with randomness r.
  - Call $\mathcal{F}^{\mathcal{R},\mathbf{TE},\mathbf{BS}}_{\mathsf{issue}}$ on input $\Big(\mathtt{ISSUE}, (\mathsf{E}_{\mathsf{RegID}}, \mathsf{K}, r, \mathsf{PK}_{\mathsf{AR}}), \mathsf{AL}, (\mathsf{IDcred}_{\mathtt{SEC}}, \mathsf{IDcred}_{\mathtt{PUB}}), \mathsf{IP}\Big)$. After receiving the response σ from $\mathcal{F}^{\mathcal{R},\mathbf{TE},\mathbf{BS}}_{\mathsf{issue}}$, set $\mathsf{cid} = (\mathsf{IP}, \mathsf{AL}, \mathsf{IDcred}_{\mathtt{SEC}}, \sigma, \mathsf{K})$.
- On input $(\mathtt{CreateACC}, \mathsf{cid}, \mathsf{P})$, proceed as follows
  - If there is no record $\mathsf{cid} = (\mathsf{IP}, \mathsf{AL}, \mathsf{IDcred}_{\mathtt{SEC}}, \sigma, \mathsf{K})$, then abort.
  - Generate an account key pair $(\mathsf{pk}_{\mathsf{ACC}}, \mathsf{sk}_{\mathsf{ACC}})$ satisfying $\mathcal{R}_{\mathsf{ACC}}$.
  - Compute $\mathsf{E}_{\mathsf{ID}} = \mathsf{TEnc}^{\mathsf{n,d}}_{\mathsf{PK}_{\mathsf{AR}}}(\mathsf{IDcred}_{\mathtt{PUB}}; \mathsf{r}')$.
  - Compute $\mathsf{RegID}_{\mathsf{ACC}} = \mathsf{PRF}_{\mathsf{K}}(x)$, where this is x'th account that is created using cid.

– Produce a NIZK π by calling $\mathcal{F}_{\mathsf{nizk}}$ for statement

$$st = (\mathsf{P}, \mathsf{E}_{\mathsf{ID}}, \mathsf{RegID}_{\mathsf{ACC}}, \mathsf{IP}, \mathsf{pk}_{\mathsf{ACC}})$$

using secret witness

$$w = (\sigma, x, r', \mathsf{IDcred}_{\mathtt{SEC}}, \mathsf{K}, \mathsf{AL}, \mathsf{sk}_{\mathsf{ACC}}, \mathsf{IDcred}_{\mathtt{PUB}})$$

for the relation $\mathcal{R}(st, w)$ that outputs $\top$ if:

1. The signature σ is valid for $(\mathsf{IDcred}_{\mathtt{SEC}}, \mathsf{K}, \mathsf{AL})$ under $\mathsf{pk}_{\mathsf{IP}}$.
2. AL satisfies the policy i.e., $\mathsf{P}(\mathsf{AL}) = \top$.
3. $\mathsf{RegID}_{\mathsf{ACC}} = \mathsf{PRF}_{\mathsf{K}}(x)$ for some $0 < x \leq \mathsf{Max}_{\mathsf{ACC}}$.
4. $\mathsf{E}_{\mathsf{ID}} = \mathsf{TEnc}^{\mathsf{n,d}}_{\mathsf{PK}_{\mathsf{AR}}}(\mathsf{IDcred}_{\mathtt{PUB}}; \mathsf{r}')$.
5. $(\mathsf{pk}_{\mathsf{ACC}}, \mathsf{sk}_{\mathsf{ACC}})$ is a valid key pair according to $\mathcal{R}_{\mathsf{ACC}}$.
6. $(\mathsf{IDcred}_{\mathtt{SEC}}, \mathsf{IDcred}_{\mathtt{PUB}})$ is a valid key pair according to $\mathcal{R}$

– Let $\mathsf{ACI} = (\mathsf{RegID}_{\mathsf{ACC}}, \mathsf{E}_{\mathsf{ID}}, \mathsf{IP}, \mathsf{pk}_{\mathsf{ACC}}, \mathsf{P}, st, \pi)$ and $\mathsf{SI}_{\mathsf{ACC}} = \mathsf{sk}_{\mathsf{ACC}}$. Send the input $(\mathtt{APPEND}, \mathsf{ACI})$ to $\mathcal{F}_{\mathsf{ledger}}$.

– Store tuple $(\mathsf{ACI}, \mathsf{SI}_{\mathsf{ACC}})$ internally and return $(\mathtt{Created}, \mathsf{RegID}_{\mathsf{ACC}})$.

- On input $(\mathtt{RETRIEVE})$, call $\mathcal{F}_{\mathsf{ledger}}$ on input $\mathtt{RETRIEVE}$. After receiving $(\mathtt{RETRIEVE}, L)$ from $\mathcal{F}_{\mathsf{ledger}}$, output L.

The protocol description for identity providers and anonymity revokers.

- On input $\mathtt{INITIALIZE}$ from $P \in \{\mathsf{IP}_1, \ldots, \mathsf{IP}_{\mathsf{m}}\}$, obtain crs from $\mathcal{F}_{\mathsf{crs}}$, generate key pair $(\mathsf{sk}_{\mathsf{IP}}, \mathsf{pk}_{\mathsf{IP}}) \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda)$ and input $(\mathtt{INITIALIZE}, (\mathsf{sk}_{\mathsf{IP}}, \mathsf{pk}_{\mathsf{IP}}))$ to $\mathcal{F}^{\mathcal{R},\mathbf{TE},\mathbf{SIG}}_{\mathsf{issue}}$.

- On input $\mathtt{INITIALIZE}$ from $P \in \{\mathsf{AR}_1, \ldots, \mathsf{AR}_{\mathsf{n}}\}$, obtain crs from $\mathcal{F}_{\mathsf{crs}}$, generate key pair $(\mathsf{sk}_{\mathsf{AR}}, \mathsf{pk}_{\mathsf{AR}}) \leftarrow\!\!\$\ \mathsf{TKeyGen}(1^\lambda)$ and input $(\mathtt{REGISTER}, \mathsf{sk}_{\mathsf{AR}}, \mathsf{pk}_{\mathsf{AR}})$ to $\mathcal{F}_{\mathsf{reg}}$.

- On input $(\mathtt{ISSUE}, \mathsf{AH})$ from identity provider IP, call $\mathcal{F}^{\mathcal{R},\mathbf{TE},\mathbf{SIG}}_{\mathsf{issue}}$ with input $(\mathtt{ISSUE}, \mathsf{AH}, \mathsf{pk}_{\mathsf{AR}})$. After receiving the response $(\mathsf{IDcred}_{\mathtt{PUB}}, \mathsf{AL}, \mathsf{E}_{\mathsf{RegID}})$ from $\mathcal{F}^{\mathcal{R},\mathbf{TE},\mathbf{SIG}}_{\mathsf{issue}}$, set $\mathsf{IPIAH} = (\mathsf{AH}, \mathsf{IDcred}_{\mathtt{PUB}}, \mathsf{AL}, \mathsf{E}_{\mathsf{RegID}})$.

- On input $(\mathtt{RETRIEVE})$, call $\mathcal{F}_{\mathsf{ledger}}$ on input $\mathtt{RETRIEVE}$. After receiving $(\mathtt{RETRIEVE}, L)$ from $\mathcal{F}_{\mathsf{ledger}}$, output L.

- On input $(\mathtt{REVOKE}, \mathsf{RegID}_{\mathsf{ACC}})$ from an identity provider IP and a set of anonymity revokers $\{\mathsf{AR}_i\}_{i \in I \subseteq [\mathsf{n}]}$, proceed as follows:

  – Anonymity revokers call $\mathcal{F}_{\mathsf{ledger}}$ on input $\mathtt{RETRIEVE}$. After receiving $(\mathtt{RETRIEVE}, L)$ from $\mathcal{F}_{\mathsf{ledger}}$, they first look up ACI in L that contains $\mathsf{RegID}_{\mathsf{ACC}}$. Next, each AR_i decrypts the E_{ID} of the ACI by computing $\mu_i = \mathsf{ShareDec}^{\mathsf{n,d}}_{\boldsymbol{pk_I}, \mathsf{sk}_i}(\mathsf{E}_{\mathsf{ID}})$. Finally, all anonymity revokers combine their shares and compute $\mathsf{IDcred}_{\mathtt{PUB}} = \mathsf{TCombine}^{\mathsf{n,d}}_{\boldsymbol{pk_I}}(\mathsf{E}_{\mathsf{ID}}, \{\mu_{\mathsf{i}}\}_{\mathsf{i} \in \mathbf{I}})$ and return $\mathsf{IDcred}_{\mathtt{PUB}}$ to the IP.

– The ACI contains the public name IP of the identity provider who registered $\mathsf{IDcred}_{\mathtt{PUB}}$. If IP is different from the requester, then ignore. Else, the IP locates the $\mathsf{IPIAH} = (\mathsf{AH}, \mathsf{IDcred}_{\mathtt{PUB}}, \mathsf{AL}, \mathsf{E}_{\mathsf{RegID}})$ record, containing the $\mathsf{IDcred}_{\mathtt{PUB}}$ that was decrypted and outputs AH.

- On input $(\mathtt{TRACE}, \mathsf{AH})$ from an identity provider IP and a set of anonymity revokers $\{\mathsf{AR}_i\}_{i \in I \subseteq [\mathsf{n}]}$, proceed as follows:
  - IP searches the database to locate the $\mathsf{IPIAH} = (\mathsf{AH}, \mathsf{IDcred}_{\mathtt{PUB}}, \mathsf{AL}, \mathsf{E}_{\mathsf{RegID}})$ containing the relevant AH and sends $\mathsf{E}_{\mathsf{RegID}}$ via $\mathcal{F}_{\mathsf{smt}}$ to the set of anonymity revokers.
  - Each AR_i computes $\mathsf{K}_i = \mathsf{ShareDec}^{\mathsf{n},\mathsf{d}}_{\boldsymbol{pk_I},\mathsf{sk}_i}(\mathsf{E}_{\mathsf{RegID}})$. Then, they call $\mathcal{F}_{\mathsf{mpc\text{-}prf}}$ on input $(\mathtt{COMPUTE}, \mathsf{K}_i)$ and receive all values $\mathsf{PRF}_{\mathsf{K}}(x)$ for $x = 1, \ldots, \mathsf{Max}_{\mathsf{ACC}}$. These values are all the possible values for $\mathsf{RegID}_{\mathsf{ACC}}$ that the AH could have used to form valid accounts.

3.5.2 Proof of Security for Identity Layer

Tolerated Corruptions. We prove security in two separate cases: with arbitrarily many malicious AHs and up to threshold semi-honest ARs, or with semi honest IP and up to threshold semi-honest ARs. Note that for technical reasons we cannot let the IP be corrupt (even if only semi-honest) at the same time with a malicious AH, since in this case the (monolithic) adversary would learn the secret key of the corrupt IP and would be able to forge invalid credentials for the corrupt AH's.

Assumptions on the environment. We consider executions with restricted adversaries and environments, that only input attribute lists AL in the ISSUE command which are valid with respect to the account holder. This restriction captures the fact that an honest IP in the real world is trusted to check (by non-cryptographic means) that an account holder AH actually satisfies the claimed attribute list AL.

Theorem 3.5.1. *Suppose that* $\mathbf{TE} = (\mathsf{TKeyGen}, \mathsf{TEnc}, \mathsf{TDec})$ *is a* (n, d)*-threshold encryption scheme, PRF is a weakly robust pseudorandom function,* $\mathcal{R}$ *a hard relation, and* $\mathbf{SIG} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{VerifySig})$ *is a EUF-CMA signature scheme, then* $\Pi_{\mathsf{id\text{-}layer}}$*, for all restricted environment (as defined above), securely implements* $\mathcal{F}_{\mathsf{id\text{-}layer}}$ *in the* $\{\mathcal{F}_{\mathsf{crs}}, \mathcal{F}_{\mathsf{reg}}, \mathcal{F}_{\mathsf{nizk}}, \mathcal{F}_{\mathsf{smt}}, \mathcal{F}^{\mathcal{R},\mathbf{TE},\mathbf{SIG}}_{\mathsf{issue}}, \mathcal{F}_{\mathsf{ledger}}, \mathcal{F}_{\mathsf{mpc\text{-}prf}}\}$*-hybrid model in the presence of an actively corrupted* AH*, and* d *semi-honest anonymity revokers* $\mathsf{AR}_1, \ldots, \mathsf{AR}_{\mathsf{d}}$*, or semi-honest* IP *and* d *semi-honest anonymity revokers* $\mathsf{AR}_1, \ldots, \mathsf{AR}_{\mathsf{d}}$.

The simulator $\mathcal{S}$ is internally emulating the functionalities $\mathcal{F}_{\mathsf{crs}}, \mathcal{F}_{\mathsf{reg}}, \mathcal{F}_{\mathsf{nizk}}, \mathcal{F}_{\mathsf{smt}}, \mathcal{F}^{\mathcal{R},\mathbf{TE},\mathbf{SIG}}_{\mathsf{issue}}$, $\mathcal{F}_{\mathsf{ledger}}, \mathcal{F}_{\mathsf{mpc\text{-}prf}}$. At the start $\mathcal{S}$ initializes empty lists $\mathtt{list}_{\mathsf{issue}}, \mathtt{list}_{\mathsf{acc}}, \mathtt{list}_{\mathsf{ledger}}, \mathtt{list}_{\mathsf{h\text{-}aid}}$, $\mathtt{list}_{\mathsf{h\text{-}pk}}, \mathtt{list}_{\mathsf{h\text{-}sig}}$. Here is the description of the simulator:

- Command INITIALIZE: $\mathcal{S}$ emulates $\mathcal{F}_{\mathsf{crs}}$ and generates public parameters for the signature scheme and threshold encryption scheme. Every time an honest or semi-honest IP or AR invokes the initialize command, the simulator generates a key-pair for them.

- Command ISSUE. *Passively corrupted* IP, *honest* AH. The simulator receives $(\texttt{ISSUE}, \mathsf{AH}, \mathsf{IP}, \mathsf{AL})$ in case of corrupt IP from the ideal functionality $\mathcal{F}_{\mathsf{id\text{-}layer}}$, and has to produce a view indistinguishable from the protocol which is consistent with this. The simulator does so by emulating the output $\mathcal{F}_{\mathsf{issue}}$ for IP namely $(\mathsf{IDcred}_{\mathtt{PUB}}, \mathsf{AL}, \mathsf{E}_{\mathsf{RegID}})$ to IP where the simulator encrypts a dummy value for $\mathsf{E}_{\mathsf{RegID}}$ and $\mathsf{IDcred}_{\mathtt{PUB}}$ is generated according to $\mathcal{R}$. The simulator adds $\mathsf{E}_{\mathsf{RegID}}$ to $\mathtt{list}_{\mathtt{h\text{-}ct}}$ and $\mathsf{IDcred}_{\mathtt{PUB}}$ to $\mathtt{list}_{\mathtt{h\text{-}pk}}$. $\mathcal{S}$ adds to $\mathtt{list}_{\mathsf{issue}}$ the entry $\langle(\mathsf{AH}, \mathsf{AL}, \mathsf{IP}); (\mathsf{E}_{\mathsf{RegID}}, \mathsf{IDcred}_{\mathtt{PUB}}, \mathsf{AL})\rangle$.

  Malicious AH*, honest* IP. When a corrupt AH invokes $\mathcal{F}_{\mathsf{issue}}^{\mathcal{R},\mathbf{TE},\mathbf{SIG}}$ on command $\Big(\texttt{ISSUE}, (\mathsf{E}_{\mathsf{RegID}}, \mathsf{K}, r, \mathsf{PK}_{\mathsf{AR}}), \mathsf{AL}, (\mathsf{IDcred}_{\mathtt{SEC}}, \mathsf{IDcred}_{\mathtt{PUB}}), \mathsf{IP}\Big)$, $\mathcal{S}$ aborts if $\mathsf{E}_{\mathsf{RegID}}$ is not an encryption of K or if $(\mathsf{IDcred}_{\mathtt{SEC}}, \mathsf{IDcred}_{\mathtt{PUB}})$ is not a valid key-pair; $\mathcal{S}$ outputs fail if $\mathsf{E}_{\mathsf{RegID}} \in \mathtt{list}_{\mathtt{h-ct}}$ or if $\mathsf{IDcred}_{\mathtt{PUB}} \in \mathtt{list}_{\mathtt{h\text{-}pk}}$.

  Otherwise, $\mathcal{S}$ calls command ISSUE of the functionality on input $(\mathsf{IP}, \mathsf{AL})$ and returns $\mathsf{cid} = (\mathsf{IP}, \mathsf{AL}, \mathsf{IDcred}_{\mathtt{SEC}}, \sigma, \mathsf{K})$ to AH where σ is a signature computed by $\mathcal{S}$ using Sign of **SIG** (since the simulator is internally emulating the honest IP w.r.t. the corrupt account holder AH). $\mathcal{S}$ adds $(\sigma; (\mathsf{IDcred}_{\mathtt{SEC}}, \mathsf{K}, \mathsf{AL}); \mathsf{IP})$ to $\mathtt{list}_{\mathtt{h\text{-}sig}}$.

- Command CreateACC. *Malicious* AH. When a corrupt AH invokes $\mathcal{F}_{\mathsf{ledger}}$ on input $\mathsf{ACI} = (st, \pi)$ (where $st = (\mathsf{P}, \mathsf{E}_{\mathsf{ID}}, \mathsf{RegID}_{\mathsf{ACC}}, \mathsf{IP}, \mathsf{pk}_{\mathsf{ACC}})$), the simulator $\mathcal{S}$ uses $\mathcal{F}_{\mathsf{nizk}}$ to extract the witness $w = (\sigma, x, r', \mathsf{IDcred}_{\mathtt{SEC}}, \mathsf{K}, \mathsf{AL}, \mathsf{sk}_{\mathsf{ACC}}, \mathsf{IDcred}_{\mathtt{PUB}})$ (or abort if the proof doesn't verify). The simulator outputs fail if one of the following condition holds: $(\sigma; (\mathsf{IDcred}_{\mathtt{SEC}}, \mathsf{K}, \mathsf{AL}); \mathsf{IP}) \notin \mathtt{list}_{\mathtt{h\text{-}sig}}$, if $\mathsf{IDcred}_{\mathtt{PUB}} \in \mathtt{list}_{\mathtt{h\text{-}pk}}$, if $\mathsf{E}_{\mathsf{ID}} \in \mathtt{list}_{\mathtt{h\text{-}ct}}$, if $\mathsf{RegID}_{\mathsf{ACC}} \in \mathtt{list}_{\mathtt{h\text{-}aid}}$, or if $x > \mathsf{Max}_{\mathsf{ACC}}$.

  Otherwise the simulator inputs the $\mathtt{CreateACC}(\mathsf{cid}, \mathsf{P}, (\mathsf{sk}_{\mathsf{ACC}}, \mathsf{pk}_{\mathsf{ACC}}))$ command to the ideal functionality and, when asked, inputs $\mathsf{aid} = \mathsf{RegID}_{\mathsf{ACC}}$ to the ideal functionality.

  Honest AH. For an honest account holder, the simulator upon receiving $(\texttt{CreateACC}, \mathsf{P}, \mathsf{pk}_{\mathsf{ACC}}, \mathsf{IP})$ from the ideal functionality, picks a random aid in the domain of the PRF and forwards it to the functionality (also adds it into $\mathtt{list}_{\mathtt{h\text{-}aid}}$). Then, the simulator prepares $st = (\mathsf{P}, \mathsf{E}_{\mathsf{ID}}, \mathsf{RegID}_{\mathsf{ACC}} = \mathsf{aid}, \mathsf{IP}, \mathsf{pk}_{\mathsf{ACC}})$ where E_{ID} is an encryption of a dummy value (and is added to $\mathtt{list}_{\mathtt{h\text{-}ct}}$), simulates a proof π via $\mathcal{F}_{\mathsf{nizk}}$ and appends (st, π) to the buffer of the ledger.

  Add entry $(\mathsf{RegID}_{\mathsf{ACC}}, \mathsf{E}_{\mathsf{ID}}, \mathsf{IP}, \mathsf{pk}_{\mathsf{ACC}}, \mathsf{P}, \pi)$ in $\mathtt{list}_{\mathtt{acc}}$.

- Command RELEASE: the simulator simulates these commands directly simulating the calls to $\mathcal{F}_{\mathsf{ledger}}$ e.g., when the adversary invokes $(\texttt{RELEASE}, \Pi)$ adds the permuted buffer to the list $\mathtt{list}_{\mathsf{ledger}}$ and then resets the buffer.

- Command RETRIEVE $\mathcal{S}$ emulates the retrieve command in $\mathcal{F}_{\mathsf{ledger}}$ and gives as output $\mathtt{list}_{\mathsf{ledger}}$.

- Command REVOKE. *Semi-honest* IP *and up to* d AR*, honest* AH. When the IP and a qualified set of AR (of which up to d are corrupt) invoke REVOKE, the simulator $\mathcal{S}$ obtains $\mathsf{RegID}_{\mathsf{ACC}} = \mathsf{aid}$ and AH from the input/output of the functionality $\mathcal{F}_{\mathsf{id\text{-}layer}}$. Now $\mathcal{S}$, using $\mathsf{RegID}_{\mathsf{ACC}}$, searches $\mathtt{list}_{\mathtt{acc}}$ and retrieves the corresponding E_{ID}. Similarly, using $(\mathsf{AH}, \mathsf{IP})$, searches $\mathtt{list}_{\mathsf{issue}}$ and retrieves the corresponding $\mathsf{IDcred}_{\mathtt{PUB}}$. Then the

simulator $\mathcal{S}$ equivocates the decryption of E_{ID} to $\mathsf{IDcred}_{\mathtt{PUB}}$ using $\mathsf{SimShare}$ (defined in the simulatability property of the threshold encryption scheme).

Malicious AH *and up to* d AR. The simulator receives $(\mathsf{RegID}_{\mathsf{ACC}}, \mathsf{AH})$ from the ideal functionality, looks up the ciphertext E_{ID} corresponding to $\mathsf{RegID}_{\mathsf{ACC}}$ and runs the threshold decryption protocol as honest parties would do.

- Command TRACE. *Semi-honest* IP *and up to* d AR*, honest* AH. When the IP and a qualified set of AR (of which up to d are corrupt) invoke TRACE, the simulator $\mathcal{S}$ receives AH and a list $\mathtt{list}_{\mathsf{RegID}_{\mathsf{ACC}}}$ of aid's from the input/output of the functionality $\mathcal{F}_{\mathsf{id\text{-}layer}}$. Now $\mathcal{S}$ recovers the ciphertext $\mathsf{E}_{\mathsf{RegID}}$. Finally $\mathcal{S}$ programs the output of $\mathcal{F}_{\mathsf{mpc\text{-}prf}}$ to be consistent with $\mathtt{list}_{\mathsf{RegID}_{\mathsf{ACC}}}$.

  Malicious AH *and up to* d AR. The simulator receives AH and a list of accounts $\{\mathsf{RegID}_{\mathsf{ACC}}\}$ from the ideal functionality. The simulator looks up the ciphertext $\mathsf{E}_{\mathsf{RegID}}$ corresponding to AH and emulates $\mathcal{F}_{\mathsf{mpc\text{-}prf}}$ to output the list $\{\mathsf{RegID}_{\mathsf{ACC}}\}$.

We now argue that the view of the environment in the real world and in the ideal world with the simulator described above are indistinguishable.

Analysis of the Simulated Game. We first argue that the probability of the simulator outputting `fail` is negligible. In the ISSUE command, since the emulation of the $\mathcal{F}_{\mathsf{issue}}$ functionality is unconditionally secure, the simulator outputs `fail` only in the following cases: 1. the adversary submits the opening for a ciphertext $\mathsf{E}_{\mathsf{RegID}}$ generated by the simulator (whose randomness is never used anywhere else, and for which the simulator only uses less than d shares of the secret key). An adversary that makes the simulation fail this way can be turned into an attack on the semantic security of the threshold encryption scheme; 2. the adversary submits the secret key for a public key $\mathsf{IDcred}_{\mathtt{PUB}}$ generated by the simulator (whose secret key is never used by the simulator). An adversary that makes the simulation fail this way can be turned into an attack onto the one-wayness of the key generation algorithm (which contradicts the assumption that $\mathcal{R}$ is a hard-relation). In the CreateACC command, since the emulation of the $\mathcal{F}_{\mathsf{nizk}}$ functionality is unconditionally secure, and since the adversary cannot make the ledger accept replayed accounts, the simulator outputs `fail` only in the following cases: 3. the adversary outputs a message/signature pair $(\sigma; (\mathsf{IDcred}_{\mathtt{SEC}}, \mathsf{K}, \mathsf{AL}); \mathsf{IP})$ which passes the verification algorithm for the public key of IP and was not generated by the simulator. An adversary that make the simulation fail this way can be turned into an attack on the unforgeability of the signature scheme; 4. the adversary inputs to the $\mathcal{F}_{\mathsf{nizk}}$ functionality the secret key for a public key $\mathsf{IDcred}_{\mathtt{PUB}}$ generated by the simulator (see bullet 2. above); 5. the adversary submits the opening for a ciphertext E_{ID} generated by the simulator (see bullet 1. above); 6. the adversary inputs to the $\mathcal{F}_{\mathsf{nizk}}$ functionality an account id $\mathsf{RegID}_{\mathsf{ACC}}$, together with the key K and input x such that $\mathsf{RegID}_{\mathsf{ACC}} = \mathsf{PRF}(\mathsf{K}, x)$ and that same $\mathsf{RegID}_{\mathsf{ACC}}$ also had been chosen at random by the simulator when simulating an honest party. An adversary that can make the simulation fail this way can be turned into an attack on the weak robust property of the PRF. 7. the adversary creates an account with $x \geq \mathsf{Max}_{\mathsf{ACC}}$. Since the $\mathcal{F}_{\mathsf{nizk}}$ is emulated with unconditional security, the probability that the simulator outputs `fail` in this way is 0.

We now slowly turn the simulated game into the real protocol via a series of hybrids.

Hybrid 0. This is the simulated protocol with the exception that we never output `fail`. As argued above, the probability that the simulator outputs `fail` is negligible, therefore the distribution generated by the simulated game and this hybrid is computationally close.

Hybrid 1. In the first hybrid we change the way in which the simulator samples the account id's aid for for honest AH. Now, instead of picking random values, the simulator picks keys K for each pair of honest account holders and certificate id $(\mathsf{AH}, \mathsf{cid})$ and computes the aid as $\mathsf{PRF}_{\mathsf{K}}(x)$ where x counts how many accounts were opened by that account holder for that cid. An adversary that can distinguish between this hybrid and the simulated protocol can be used to break the pseudorandomness property of PRF.

Hybrid 2. In the second hybrid we change the way in which the simulator generates the ciphertexts E_{ID} for the honest parties during the simulation of `CreateACC`. Now the simulator encrypts the value $\mathsf{IDcred}_{\mathtt{PUB}}$ corresponding to the certificate cid of the account holder AH. An adversary that can distinguish between this hybrid and the previous one can be used to break the semantic security of the threshold encryption scheme.

Hybrid 3. In the third hybrid we change the way in which the simulator generates the ciphertexts $\mathsf{E}_{\mathsf{RegID}}$ for the honest parties during the simulation of `ISSUE`. Now the simulator encrypts the value K corresponding to the certificate cid of the account holder AH, consistently with the value of K which had been sampled for that cid in hybrid 1. An adversary that can distinguish between this hybrid and the previous one can be used to break the semantic security of the threshold encryption scheme.

Hybrid 4. In the fourth hybrid we change the behaviour of the simulator during `REVOKE` for honest AH. Now we let the simulator run the threshold decryption protocol honestly. Note that, since we changed the encryptions $\mathsf{E}_{\mathsf{RegID}}$ in a previous hybrid to encrypt the right value, the output of the decryption in this and the previous hybrid is identical. Hence, an adversary that can distinguish between this and the previous hybrid can be used to break the partial decryption simulatability (PDS) of the threshold encryption scheme.

Hybrid 5. In this hybrid we change the behaviour of the simulator during `TRACE` for honest AH. Now we let the simulator run the code of $\mathcal{F}_{\mathsf{mpc\text{-}prf}}$ honestly instead of programming the output of the functionality. Note that, since we changed the encryptions E_{ID} in a previous hybrid to encrypt the right value, the output of the decryption in this and the previous hybrid is identical. Since the simulation of $\mathcal{F}_{\mathsf{mpc\text{-}prf}}$ is perfect, this hybrid is identically distributed to the previous one.

Hybrid 6. In the final hybrid we change the behaviour of the simulator when simulating the proofs π for honest account holders AH. Up until now the simulator had simply picked random strings π, stored them together with the statement x, and then emulated the answers to `VERIFY` queries of the adversary for such proofs by outputting 1 on behalf of the $\mathcal{F}_{\mathsf{nizk}}$ functionality. Now instead the simulator runs the code of the functionality $\mathcal{F}_{\mathsf{nizk}}$, that is it only outputs that a proof is valid if it can come up with a witness for it. Note that, since we changed all elements in the statement (the encryption, the account id, etc.) in previous hybrids to match the behaviour

of a honest account holder, the simulator knows the necessary witnesses, thus the previously simulated proofs still verify, and this hybrid is identically distributed to the previous one.

Since hybrid 6 is identical to the real protocol (that is, the real protocol run in the hybrid world where all the helping functionalities are available), this concludes the proof.

3.5.3 Credential Issue Protocol

The issue protocol Π_{issue} uses as its main ingredient a two-round blind signature scheme (as defined in Section 3.2.4), augmented with a NIZK that proves that the input to the blind-signature protocol is consistent with the ciphertext and the public-key that the account holder sends to the identity provider.

Protocol Issue Π_{issue}

The protocol operates in the $\{\mathcal{F}_{\mathsf{crs}}, \mathcal{F}_{\mathsf{reg}}, \mathcal{F}_{\mathsf{nizk}}, \mathcal{F}_{\mathsf{smt}}\}$-hybrid model. Let $\mathbf{BS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}_1, \mathsf{Sign}_2, \mathsf{Unblind}, \mathsf{VerifySig})$ be a blind signature scheme and $\mathbf{TE} = (\mathsf{TKeyGen}, \mathsf{TEnc}, \mathsf{TDec})$ be a (n, d)-threshold encryption scheme, and $\mathcal{R}$ an NP relation corresponding to the account holder's key pair.

- Upon input $(\texttt{SETUP})$, use $\mathcal{F}_{\mathsf{crs}}^{\mathsf{Setup}}$ to generate $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ and publicize it to all parties.
- Upon input $(\texttt{INITIALIZE}, (\mathsf{sk}_{\mathsf{IP}}, \mathsf{pk}_{\mathsf{IP}}))$, the identity provider IP checks if the key pair has a correct distribution with respect to the $\mathsf{KeyGen}(\mathsf{pp})$. If yes, stores $(\mathsf{sk}_{\mathsf{IP}}, \mathsf{pk}_{\mathsf{IP}})$ and sends $(\texttt{REGISTER}, \mathsf{sk}_{\mathsf{IP}}, \mathsf{pk}_{\mathsf{IP}})$ to $\mathcal{F}_{\mathsf{reg}}$.
- Upon an input $\Big(\texttt{ISSUE}, (ct, m, r, \mathsf{pk}_{\mathsf{AR}}), \mathsf{aux}, (\mathsf{sk}_{\mathsf{AH}}, \mathsf{pk}_{\mathsf{AH}}), \mathsf{IP}\Big)$ to the account holder AH and an input $(\texttt{ISSUE}, \mathsf{AH}, \mathsf{pk}_{\mathsf{AR}})$ to an identity provider IP, proceed as follows:
  1. AH retrieves $\mathsf{pk}_{\mathsf{IP}}$ from $\mathcal{F}_{\mathsf{reg}}$, computes $\texttt{sign}_1 = \mathsf{Sign}_1(\mathsf{pk}_{\mathsf{IP}}, (\mathsf{sk}_{\mathsf{AH}}, m, \mathsf{aux}), \mathsf{pp}; r')$ and sends $(\texttt{PROVE}, st, w)$ to $\mathcal{F}_{\mathsf{nizk}}$ for statement $st = (\mathsf{pk}_{\mathsf{AH}}, \texttt{sign}_1, ct, \mathsf{aux}, \mathsf{pp})$ using secret witness $w = (\mathsf{sk}_{\mathsf{AH}}, m, r', r)$ for the relation $\mathcal{R}_1(st, w)$ that outputs $\top$ if:
     a) $(\mathsf{sk}_{\mathsf{AH}}, \mathsf{pk}_{\mathsf{AH}}) \in \mathcal{R}$.
     b) $\texttt{sign}_1 = \mathsf{Sign}_1(\mathsf{pk}_{\mathsf{IP}}, (\mathsf{sk}_{\mathsf{AH}}, m, \mathsf{aux}), \mathsf{pp}; r')$.
     c) $ct = \mathsf{TEnc}^{\mathsf{n},\mathsf{d}}_{\mathsf{PK}_{\mathsf{AR}}}(m; r)$
  2. Upon receiving the proof π, AH sends $(\texttt{SEND}, \mathsf{IP}, (st, \pi))$ to $\mathcal{F}_{\mathsf{smt}}$.
  3. Upon receiving $(\texttt{SENT}, \mathsf{AH}, (st, \pi))$ from $\mathcal{F}_{\mathsf{smt}}$, IP inputs $(\texttt{VERIFY}, st, \pi)$ (for relation $\mathcal{R}_1$) to $\mathcal{F}_{\mathsf{nizk}}$. If they pass, IP computes and sends (through $\mathcal{F}_{\mathsf{smt}}$) $\texttt{sign}_2 \leftarrow \mathsf{Sign}_2(\mathsf{sk}_{\mathsf{IP}}, \texttt{sign}_1)$.
     AH runs $\mathsf{Unblind}(\texttt{sign}_2, r')$ and obtains a signature σ on $(\mathsf{sk}_{\mathsf{AH}}, m, \mathsf{aux})$.

3.5.4 Proof of Security for Issue Protocol

Theorem 3.5.2. *Assume that* $\mathbf{BS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}_1, \mathsf{Sign}_2, \mathsf{Unblind}, \mathsf{VerifySig})$ *is a blind signature scheme. Then,* Π_{issue} *securely implements* $\mathcal{F}_{\mathsf{issue}}$ *in the* $\{\mathcal{F}_{\mathsf{crs}}, \mathcal{F}_{\mathsf{reg}}, \mathcal{F}_{\mathsf{nizk}}, \mathcal{F}_{\mathsf{smt}}\}$*-hybrid model in the presence of an actively corrupted* AH *or a passively corrupted* IP.

Proof. We consider corruptions of AH and IP separately.
(Actively) Corrupt AH. The simulator for a corrupt AH receives (st, π) from the $\mathcal{F}_{\mathsf{smt}}$ functionality and extracts the witness w from $\mathcal{F}_{\mathsf{nizk}}$ (or abort if the proof doesn't verify). Now the simulator can input the `ISSUE` command to the ideal functionality which includes the message and randomness used to generate ct, the auxiliary information, and the account public and secret keys. The simulator therefore receives a signature σ from the ideal functionality. Using the simulator guaranteed from the simulatability property (Definition 3.2.7), the simulator computes the appropriate $\mathtt{sign}_2$ message. Note that we can use this simulator since (due to the soundness of the NIZK) we know that $\mathtt{sign}_1$ was computed according to the protocol and our simulator knows the message m and the randomness r' which is used by the adversary to do so.

We claim that this simulation is indistinguishable from the real protocol. To see why, notice that the output of the IP is identical in the real and ideal world (remember that in the $\mathcal{F}_{\mathsf{nizk}}$ hybrid model the proofs are unconditionally sound). Moreover, the view of the corrupted AH consists only of the message $\mathtt{sign}_2$. Therefore, an adversary that can distinguish between the real and simulated protocol can directly be turned into an adversary that breaks the simulatability property (Definition 3.2.7) of the blind signature scheme.
(Passively) Corrupt IP. The case of a corrupted IP is relatively simple since we only consider passively corrupt IP: Here the simulator gets the input and output of the IP from the ideal functionality (which consists of the account public key, the ciphertext, and the auxiliary information), and has to produce a view indistinguishable from the protocol which is consistent with this output. The simulator does so by computing the first message of the blind signature $\mathtt{sign}_1$ by running Sign_1 with some dummy input. Note that all other parts of the statement st are at this point known to the simulator, which therefore only needs to simulate the proof π, but this is trivial to do in the $\mathcal{F}_{\mathsf{nizk}}$-hybrid model by simply appending an arbitrary proof π^* to the view, and then answering verify queries so that π^* is a valid proof for the statement st.

We claim that this simulation is indistinguishable from the real protocol. To see why, notice that the only difference in the view of IP between the real and simulated protocol is in the distribution of $\mathtt{sign}_1$ and therefore an adversary that can distinguish between the real and simulated protocol can directly be turned into an adversary that breaks the blindness property (Definition 3.2.6) of the blind signature scheme.

Finally, note that at first glance it might appear strange that the proof does not mention at all the security of the threshold encryption scheme. However, this is because our ideal functionality just guarantees that the ciphertext is correctly computed according to the encryption algorithm. Thus, our protocol securely implements the functionality regardless of which security (if any) is guaranteed by the encryption scheme!

□

3.6 Implementation Details

This section contains a detailed description of the protocols to implement for the ID layer.

3.6.1 Auxiliary Σ-protocols.

Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, g_2)$ be a bilinear group as defined in 2.6.2. In the Σ protocols described in the following we assume that all group elements and commitment keys and public keys for signature and encryption schemes come from a group of order p, where in some cases it does not matter if we use $\mathbb{G}_1$, $\mathbb{G}_2$ or $\mathbb{G}_T$, but there are also cases where a particular group has to be used. The protocol applying a Σ-protocol will specify which group is used. We denote by $\mathsf{PK}\{(x, y, \cdots) : \text{ statements about } x, y\}$, a zero knowledge proof of knowledge of $x, y, \cdots$ that satisfies statements. Here, $x, y, \cdots$ are private (witness), and other values in statements are public. The sigma protocols we describe below are standard protocols that have appeared in other works [42, 151] and follow from the *pre-image protocol* of Maurer [133] for proving knowledge of a pre-image of a group homomorphism that unifies and generalizes a large class of protocols in literature.

3.6.2 Using Fiat-Shamir for non-interactive Proof

All Σ protocols below are described as interactive 2-party protocols. When using them for non-interactive proofs, we use the Fiat-Shamir paradigm with hash function $\mathcal{H}$ to replace a random oracle. More precisely, this will be done as follows: Suppose the messages in the protocol are (a, c, z), where c is the random challenge from the verifier, and where x is the public input. Then the prover executes the protocol on his own as follows: they compute a first, and then $\mathcal{H}(x, a)$. They set c to be the first $\lfloor \log_2 p \rfloor$ bits of the hash value. Finally they compute the last message z. The non-interactive proof is (c, z).

To verify a proof we use the fact that all our Σ protocols have a verification equation that allows you to compute a from x, c and z. So given x and e, z, the verifier computes a, $\mathcal{H}(x, a)$ and compares c to the first $\lfloor \log_2 p \rfloor$ bits of the hash value.

If a proof consists of several, say T Σ protocols, we do them all in parallel as follows: the prover computes all T first messages and concatenates them to get M_1. Then they compute $\mathcal{H}(X, M_1)$ where X is the concatenation of all public inputs to the protocols. They set c to be the first $\lfloor \log_2 p \rfloor$ bits of the hash value. Finally they compute the T last messages and concatenates them all to get M_2. The non-interactive proof is (c, M_2).

To verify X and (c, M_2), where X contains all the public inputs, the verifier splits M_2 into T individual last messages. As above, they use the verification equations to compute the T first messages. They concatenate them to get M_1, computes $\mathcal{H}(X, M_1)$ and sets c to the first $\lfloor \log_2 p \rfloor$ bits of the hash value.

3.6.3 Sigma protocol for proving knowledge of discrete log

Protocol dlog : $\mathsf{PK}\{(x) : y = g^x\}$

- The prover computes and sends $a = g^\alpha$ for randomly chosen $\alpha \in \mathbb{Z}_p^*$.
- The verifier sends a random challenge c at random from $\mathbb{Z}_p$.
- The prover sends $z = \alpha + cx \mod p$
- The verifier checks if $a = g^{-z}y^c$. If yes, the verifier accepts.

We describe here for completeness the non-interactive variant using Fiat-Shamir but remember that this approach will be used in parallel for several protocols for applications that require more than one Sigma protocol.

Protocol NI dlog : $\mathsf{PK}\{(x) : y = g^x\}$

- The prover computes $a = g^\alpha$ for randomly chosen $\alpha \in \mathbb{Z}_p^*$, $c = H(y||a)$, $z = \alpha + cx$ and sends (c, z) to the verifier.
- The verifier computes if $a = y^{-c}g^z$ and checks that $c = H(y||a)$. If yes, the verifier accepts.

3.6.4 Sigma protocol for proving equality of committed value and Elgamal encrypted value

The prover has produced an ElGamal encryption $(e_1, e_2) = (g_1^R, g_1^x h_1^R)$ under public key g_1, h_1 that is an encryption of g_1^x with randomness R. They have also committed to x: $C = g^x h^r$ and wants to show that the same x appears in the encryption and in the commitment.

Protocol com−enc−eq : $\mathsf{PK}\{(x, r, R) : e_1 = g_1^R \wedge e_2 = g_1^x h_1^R \wedge C = g^x h^r\}$

1. The prover computes $a_1 = g_1^\alpha, a_2 = g_1^\beta h_1^\alpha, a_3 = g^\beta h^\gamma$ for randomly chosen $\alpha, \beta, \gamma \in \mathbb{Z}_p^*$ and sends (a_1, a_2, a_3) to the verifier.
2. The verifier sends a random challenge c at random from $\mathbb{Z}_p$.
3. The prover computes $z_1 = \alpha + cR \mod p$, $z_2 = \beta + cx \mod p$, $z_3 = \gamma + cr \mod p$ and sends (z_1, z_2, z_3) to the verifier.
4. The verifier checks if $a_1 = g_1^{-z_1}e_1^c, a_2 = g_1^{-z_2}h_1^{-z_1}e_2^c, a_3 = g^{-z_2}h^{-z_3}C^c$. If yes, the verifier accepts.

3.6.5 Proof of Equality of Aggregated Discrete Logs & Commitments

We now describe a protocol for proving equality of the discrete logarithms $(a_1, \ldots, a_n)$ in $y = \prod_{i=1}^n G_i^{a_i}$ and individual algebraic commitments to them. Using standard notation, we denote

the protocol by $\mathsf{PK}\{(a_1, \ldots, a_n, r_1, \ldots, r_n) : y = \prod_{i=1}^{n} G_i^{a_i} \wedge C_1 = g^{a_1} h^{r_1} \wedge \cdots \wedge C_n = g^{a_n} h^{r_n}\}$.

Let $\mathbb{G}$, $\mathbb{H}$ be groups of prime order p. Given $y = \prod G_i^{a_i}$ and $C_i = g^{a_i} h^{r_i}$, where G_i are generators of the group $\mathbb{G}$, g is a generator of $\mathbb{H}$ and h is a random element in $\mathbb{H}$. The prover does not know the discrete logarithm of h with respect to g, and the discrete logarithms of G_is with respect to each other. We want to prove equality of the discrete logarithms in y and the respective values committed to in C_is.

Protocol comEq : $\mathsf{PK}\{(a_1, \ldots, a_n, r_1, \ldots, r_n) : y = \prod_{i=1}^{n} G_i^{a_i} \wedge C_1 = g^{a_1} h^{r_1} \wedge \cdots \wedge C_n = g^{a_n} h^{r_n}\}$

Given $y = \prod_{i=1}^{n} G_i^{a_i}$ and $C_i = g^{a_i} h^{r_i}$

1. The prover computes the following values: $u = \prod_{i=1}^{n} G_i^{\alpha_i}$ and $v_i = g^{\alpha_i} h^{R_i}$ for randomly chosen $\alpha_i, R_i \in \mathbb{Z}_p$ and sends u, v_i to the verifier.
2. The verifier chooses a challenge c at random from $\mathbb{Z}_{2^k}$ for a fixed k, such that $2^k < p$, and sends it to the prover.
3. For a challenge string c, prover computes and sends the tuple (s_i, t_i)

$$s_i = \alpha_i - ca_i \pmod{p}, \qquad t_i = R_i - cr_i \pmod{p}$$

4. Verification: Check if $u = y^c \prod G_i^{s_i}$ and $v_i = (C_i)^c g^{s_i} h^{t_i}$. The verifier accepts if checks succeed for all i.

3.6.6 Proof of Aggregated Discrete Logs

This is a specialization of the previous protocol and proves knowledge of discrete logarithms $(a_1, \ldots, a_n)$ in $y = \prod_{i=1}^{n} G_i^{a_i}$. Protocol is denoted by $\mathsf{PK}\{(a_1, \ldots, a_n) : y = \prod_{i=1}^{n} G_i^{a_i}\}$. The protocol is derived by simply ignoring in the previous protocol everything related to the commitments C_i.

Protocol AggregateDL : $\mathsf{PK}\{(a_1, \ldots, a_n) : y = \prod_{i=1}^{n} G_i^{a_i}\}$

Given $y = \prod_{i=1}^{n} G_i^{a_i}$

1. The prover computes the following values: $u = \prod_{i=1}^{n} G_i^{\alpha_i}$ for randomly chosen $\alpha_i \in \mathbb{Z}_p$ and sends u to the verifier.
2. The verifier chooses a challenge c at random from $\mathbb{Z}_{2^k}$ for a fixed k, such that $2^k < p$, and sends it to the prover.
3. For a challenge string c, prover computes and sends the tuple (s_i)

$$s_i = \alpha_i - ca_i \pmod{p}$$

4. Verification: Check if $u = y^c \prod G_i^{s_i}$. The verifier accepts if checks succeed for all i.

3.6.7 Proof of Equality for Commitments in Different Groups

This is essentially a specialization of the protocol from section 3.6.5. Assume we have discrete logarithms (a_1, a_2) in $y = G_1^{a_1} G_2^{a_2}$ which can be seen as a commitment to a_1 with randomness a_2, as well as a commitment to a_1 in a different group. Using standard notation, we denote the protocol by $\mathsf{PK}\{(a_1, a_2, r) : y = G_1^{a_1} G_2^{a_2} \wedge C = g^{a_1} h^r\}$.

Let $\mathbb{G}$, $\mathbb{H}$ be groups of prime order p. Given $y = G_1^{a_1} G_2^{a_2}$ and $C = g^{a_1} h^r$, where G_i are generators of the group $\mathbb{G}$, g is a generator of $\mathbb{H}$ and h is a random element in $\mathbb{H}$. The prover does not know the discrete logarithm of h with respect to g, and the discrete logarithms of G_is with respect to each other. We want to prove that y and C are commitments to the same value.

Protocol $\mathsf{PK}\{(a_1, a_2, r) : y = G_1^{a_1} G_2^{a_2} \wedge C = g^{a_1} h^r\}$

Given $y = G_1^{a_1} G_2^{a_2}$ and $C = g^{a_1} h^r$

1. The prover computes the following values: $u = G_1^{\alpha_1} G_2^{\alpha_2}$ and $v = g^{\alpha_1} h^R$ for randomly chosen $\alpha_i, R \in \mathbb{Z}_p$ and sends u, v to the verifier.
2. The verifier chooses a challenge c at random from $\mathbb{Z}_{2^k}$ for a fixed k, such that $2^k < p$, and sends it to the prover.
3. For a challenge string c, prover computes and sends the tuple (s_1, s_2, t), computed as follows:

$$s_1 = \alpha_1 - ca_1 \pmod{p},\ s_2 = \alpha_2 - ca_2 \pmod{p},\ t = R - cr \pmod{p}$$

4. Verification: Check if $u = y^c \prod G_1^{s_1} G_2^{s_2}$ and $v = C^c g^{s_1} h^t$. The verifier accepts if all checks succeed.

3.6.8 Proof of multiplicative relation on committed values

This protocol proves, for committed values a_1, a_2, a_3 in commitments C_1, C_2, C_3, that $a_1 a_2 = a_3 \bmod p$. Protocol is denoted by $\mathsf{PK}\{(a_1, a_2, a_3, r_1, r_2, r_3) : C_i = g^{a_i} h^{r_i} \text{ for } i = 1..3 \wedge a_1 a_2 = a_3 \bmod p\}$.

Protocol comMult : $\mathsf{PK}\{(a_1, a_2, a_3, r_1, r_2, r_3) : C_i = g^{a_i} h^{r_i} \text{ for } i = 1..3 \wedge a_1 a_2 = a_3 \bmod p\}$

Given $C_i = g^{a_i} h^{r_i}$ for $i = 1, 2, 3$, we will prove the relation $a_1 a_2 = a_3 \bmod p$ by proving that $C_3 = C_1^{a_2} h^r$ in parallel with proving that $C_i = g^{a_i} h^{r_i}$ for $i = 1, 2, 3$. The condition is satisfied if we set $r = r_3 - r_1 a_2 \bmod p$, so the prover can do this.
If the proof succeeds, it follows that C_3 can be opened to both a_3 and $a_1 a_2$, so we get what we want unless the binding condition is broken.

1. The prover computes the following values: $v_i = g^{\alpha_i} h^{R_i}$ for $i = 1, 2, 3$, $v = C_1^{\alpha_2} h^R$ for randomly chosen $\alpha_i, R_i, \alpha, R \in \mathbb{Z}_p$ and sends $\{v_i\}, v$ to the verifier.

   Note that v will later be used in a verification equation to test $C_3 = C_1^{a_2} h^r$. This is

why we need to use α_2 in the exponent of C_1 in the expression for v.

2. The verifier chooses a challenge c at random from $\mathbb{Z}_{2^k}$ for a fixed k, such that $2^k < p$, and sends it to the prover.

3. For a challenge string c, prover computes and sends the tuple (s_i, t_i) for $i = 1, 2, 3$ and t:

$$s_i = \alpha_i - ca_i \pmod{p},\ t_i = R_i - cr_i \pmod{p},\ t = R - cr \pmod{p}$$

4. Verification: Check if $v_i = (C_i)^c g^{s_i} h^{t_i}$ for $i = 1, 2, 3$ and $v = C_3^c C_1^{s_2} h^t$. The verifier accepts if all checks succeed.

3.6.9 Protocol for Π_{issue}

Protocol For proving that $ct = \mathsf{TEnc}^{\mathsf{n},\mathsf{d}}_{\mathsf{PK}_{\mathsf{AR}}}(m; r)$

Recall that we follow the share-and-encrypt paradigm for the threshold encryption by using Shamir Secret sharing and the CL encryption scheme. This means that the encryption of m is supposed to be consisting of n ciphertexts $\mathsf{CL.Enc}(\mathsf{pk}_{\mathsf{AR}_i}, \mathsf{sh}(m)_i)$ for $i = 1, \ldots, \mathsf{n}$ where $\mathsf{sh}(m)_i$ is the i'th share of m.
Let M' be the commitment of m on group $\widetilde{\mathbb{G}}$. The prover AH proves to the verifier IP that M' contains the same value m as does ct:

1. AH makes a commitment B_0 to m under the default commitment key ck (which is in the CRS), so $B_0 \in \mathbb{H}$. They use the protocol from Section 3.6.7 to show that M' and B_0 both contain m.

2. AH establishes a commitment M_i to each share $\mathsf{sh}(m)_i$ of m as follows: when AH secret-shares m, they use a polynomial g to get shares $\mathsf{sh}(m)_i$. Let the coefficients of g be $b_0 = m, b_1, ..., b_{\mathsf{d}}$. Note that we have $\mathsf{sh}(m)_i = g(i) = \sum_{j=0}^{\mathsf{d}} b_j \cdot i^j$. Now, the AH makes and sends commitments B_j to b_j, where we notice that B_0 has already been constructed above. Using the homomorphic property of commitments one can compute $K_i = \prod_{j=0}^{\mathsf{d}} B_j^{i^j}$ which is a commitment to $\mathsf{sh}(m)_i$. Since AH, the prover, has created the B_i's, they also know how to open the K_i's.

3. AH finally runs a protocol similar to $\mathsf{com{-}enc{-}eq}$ from section 3.6.4 but for the CL encryption scheme to show the IP that each $\mathsf{CL.Enc}(\mathsf{pk}_{\mathsf{AR}_i}, \mathsf{sh}(m)_i)$ contains the same value as K_i for $i = 1...\mathsf{n}$.

3.6.10 Protocol for $\Pi_{\mathsf{id-layer}}$

Proving that you know a signature. The user has a signature $\sigma = (a, b)$ on a message $(m_0, ..., m_\ell)$ and wants to convince a verifier of this fact. This is done as follows:

Protocol For proving knowledge of a signature

1. Choose $r, r' \leftarrow \mathbb{F}_p$, and compute a blinded version of the signature as follows:

$$\hat{a} = a^r, \hat{b} = (b \cdot a^{r'})^r$$

   and sends the blinded signature $(\hat{a}, \hat{b})$ to the verifier.

2. Both parties compute locally the following values:

$$v_1 = \hat{e}(\hat{a}, g_2), v_2 = \hat{e}(\hat{b}, g_2), v_3 = \hat{e}(\hat{a}, \tilde{X}), u_i = \hat{e}(\hat{a}, \tilde{Y_i})$$

3. The verifier checks that $\hat{a} \neq 1_{\mathbb{G}_1}$. If not, the verifier rejects. The user gives a ZK proof of knowledge:

$$PK((m_0, ..., m_\ell, r') : v_2 = v_3 \cdot v_1^{r'} \prod_{i=1}^{\ell} u_i^{m_i})$$

4. The verifier accepts if the proof is valid.

Adaptation for Implementing $\Pi_{\mathsf{id-layer}}$. In the ID layer, the AH will act as user/prover and anyone else can act as verifier. However, the above protocol needs to be further adapted so we can show specific properties of attributes: recall that we set $(m_0, ..., m_\ell) = (r, \mathsf{IDcred}_{\mathsf{SEC}}, \mathsf{K}, a_1, ..., a_v)$, padding the right-hand side with 0's if needed to get ℓ entries. The AH can supply also commitments to all fields and attributes: $i = 1...\ell : C_i = \mathsf{Commit}_{\mathsf{ck}}(m_i, r_i)$. So we execute the protocol exactly as specified above, but the proof of knowledge in step 3 is extended as follows:

$$PK((m_0, ..., m_\ell, r') : v_2 = v_3 \cdot v_1^{r'} \prod_{i=1}^{\ell} u_i^{m_i},\ C_i = \mathsf{Commit}_{\mathsf{ck}}(m_i, r_i) \text{ for } i = 1...\ell).$$

For this proof of knowledge, we use the comEq protocol from section 3.6.5-
The AH can now use the C_i to prove desired properties of the fields and attributes.

Protocol For proving that $\mathsf{RegID}_{\mathsf{ACC}} = \mathsf{PRF}_{\mathsf{K}}(x)$ for some $x \leq \mathsf{Max}_{\mathsf{ACC}}$

Recall that $\mathsf{PRF}_{\mathsf{K}}(x) = \bar{g}^{1/(x+\mathsf{K})}$. This value can be seen as $\mathsf{Commit}_{\mathsf{ck}}(1/(\mathsf{K}+x))$ where the randomness is 0. Let $C_1 = \mathsf{Commit}_{\mathsf{ck}}(\mathsf{K})$ and $C_3 = \mathsf{Commit}_{\mathsf{ck}}(a_1) = \mathsf{Commit}_{\mathsf{ck}}(\mathsf{Max}_{\mathsf{ACC}})$.

1. AH makes commitments $F_x = \mathsf{Commit}_{\mathsf{ck}}(x)$ and $F_1 = g$ which can be written as $\mathsf{Commit}_{\mathsf{ck}}(1)$ where the randomness is 0. Note that $C_1 F_x = \mathsf{Commit}_{\mathsf{ck}}(\mathsf{K} + x)$.

2. AH runs Protocol comMult from Section 3.6.8 with input commitments $C_1 F_x$, $\mathsf{RegID}_{\mathsf{ACC}}$ and F_1. This shows that $\mathsf{RegID}_{\mathsf{ACC}} = \mathsf{PRF}_{\mathsf{K}}(x)$.

3. AH finally uses a zkSNARK on committed inputs based on F_x and C_3 to show that $x \leq \mathsf{Max}_{\mathsf{ACC}}$.

Protocol For proving that $\mathsf{E}_{\mathsf{ID}} = \mathsf{TEnc}^{\mathsf{n,d}}_{\mathsf{PK}_{\mathsf{AR}}}(\mathsf{IDcred}_{\mathtt{PUB}}; \mathsf{r}')$

Let C_0 be the commitment to $\mathsf{IDcred}_{\mathtt{SEC}}$. Recall that we follow share-and-encrypt paradigm for the threshold encryption (see 3.2.6) which means in order to encrypt $\mathsf{IDcred}_{\mathtt{PUB}}$ correctly, AH first secret shares $\mathsf{IDcred}_{\mathtt{SEC}}$ using a polynomial f to get shares $\mathsf{sh}(\mathsf{IDcred}_{\mathtt{SEC}})_{\mathsf{i}}$ and then computes the i'th component of E_{ID} as Elgamal encryption $\mathsf{Enc}_{\mathsf{pk}_{\mathsf{AR}_i}}(\bar{g}^{\mathsf{sh}(\mathsf{IDcred}_{\mathtt{SEC}})_{\mathsf{i}}}; r'_i)$. Let the coefficients of f be $a_0 = \mathsf{IDcred}_{\mathtt{SEC}}, a_1, ..., a_{\mathsf{d}}$. Note that we have $\mathsf{sh}(\mathsf{IDcred}_{\mathtt{SEC}})_{\mathsf{i}} = \mathsf{f}(\mathsf{i}) = \sum_{\mathsf{j}=0}^{\mathsf{d}} \mathsf{a}_{\mathsf{j}} \cdot \mathsf{i}^{\mathsf{j}}$. Now,

1. AH makes commitments A_j to a_j, where we set $A_0 = C_0$. Using the homomorphic property of commitments one can compute $S_i = \prod_{j=0}^{\mathsf{d}} A_j^{i^j}$ which is a commitment to $\mathsf{sh}(\mathsf{IDcred}_{\mathtt{SEC}})_{\mathsf{i}}$. Since AH, the prover, has created the A_i's, they also know how to open the S_i's.

2. For $i = 1...\mathsf{n}$, the AH runs the protocol com$-$enc$-$eq from section 3.6.4 using the two components in $E_{\mathsf{pk}_{\mathsf{AR}_i}}(\bar{g}^{\mathsf{sh}(\mathsf{IDcred}_{\mathtt{SEC}})_{\mathsf{i}}}; r'_i)$ as e_1, e_2 and S_i as C.

3.7 Putting Everything Together

We presented all components of the system in a modular way. We now describe how to instantiate each of the components needed in the ID-layer.

UC-NIZK. We use two different types of non-interactive zero knowledge proofs in our implementation. One is based on Σ-protocols made non-interactive with the Fiat-Shamir (FS) transform [84], and the other is preprocessing-based zkSNARKs [99, 141] in the crs model. Unfortunately, known instantiations of both types of NIZKs do not satisfy UC-security.

In order to lift SNARK to be UC-secure we use the transformation of Kosba et al. [127]. At a high level, the transformation works having the prover prove an augmented relation $\mathcal{R}_{\mathcal{L}'}$ as follows: a pair of one-time signing/verification keys are generated for each proof. The prover is additionally required to show that a ciphertext encrypts the witness of the underlying relation $\mathcal{R}_{\mathcal{L}}$, or the PRF was correctly evaluated on the signature key under a committed key. Then the prover is required to sign the statement together with the proof of $\mathcal{L}'$. Since our goal is to use SNARKs on small circuits for the purposes of prover efficiency, we treat the augmented relation $\mathcal{R}_{\mathcal{L}'}$ as a composite statement [9, 48] and use a combination of SNARKs and sigma protocols to prove the augmented relation of the transformation. We use the CL scheme [54] for encryption, and $f_k : x \to H(x)^k$, for $k \in \mathbb{Z}_{|}$ as the PRF where H maps bit strings to group elements. This PRF can be shown to be secure under the DDH assumption where H is modeled as a random oracle. We can use a sigma protocol to prove correct evaluation of the PRF given public input, public output, and committed key g^k. A standard sigma protocol proof of equality

of discrete logarithms can be used to prove equality of CL encrypted and Pedersen committed messages. The composition theorem from [9] can be invoked to argue security of the NIZK for the composite statement formed as the AND of the statements of the lifting transformation.

In [70], we show that a simulation-sound NIZK (such as Fiat-Shamir as shown in [81]) and a perfectly correct CPA-secure encryption scheme are sufficient to instantiate a simulation-extractable NIZK by transforming the relation to include a ciphertext encrypting the witness.[6]

While this lifting technique for transforming a (sound) NIZK to a knowledge-sound NIZK is folklore, the use of CL encryption scheme [54] for this goal is novel up to our knowledge. Our choice of the CL encryption scheme in the transformation means that we can at the same time encrypt messages in the same plaintext space as the commitment schemes (thus allowing for efficient proofs of equality of discrete logarithms), and guarantee efficient decryption by the extractor. This is as opposed to using e.g., Paillier (where we could have efficient decryption but would need range proofs to prove equality of exponents in different groups) or ElGamal "in the exponent" (where the group order could be the same but efficient decryption can only be achieved by encrypting the witness in short chunks).

Implementation of Π_{issue}. We instantiate the blind signature scheme $\mathbf{BS}$ by the Pointcheval-Sanders (PS) signature scheme [145]. We recall the PS scheme in Section 3.2.4, and prove that the PS signature satisfies the definitions of Blindness 3.2.6 (here we prove a stronger variant than what given in the original paper) and Simulatability 3.2.7 (which we define, as it is needed for proving UC security of the overall construction). We adopt the threshold encryption $\mathbf{TE}$ scheme (Definition 3.2.12) described in [74] which follows the share and encrypt paradigm. We use the CL encryption scheme [54] to encrypt. Once again, our choice of CL encryption scheme means that we can at the same time encrypt messages in the same plaintext space as the commitment schemes (for efficient equality proofs) and guarantee efficient decryption when needed in the TRACE command by the ARs.

We now describe the Σ-protocols we use to prove relation $\mathcal{R}_1$ in Π_{issue}. We let $\mathcal{R}$ be the discrete log relation where $\mathsf{pk} = g^{\mathsf{sk}}$. Then, we can prove that public keys and secret keys satisfy $\mathcal{R}$ using the standard Σ-protocol dlog from Section 3.6.3. The message output by $\mathtt{Sign}_1$ in the PS blind signature is essentially a Pedersen commitment for vectors (see Section 3.2.4). So we prove that $\mathtt{Sign}_1$ was executed correctly using the AggregateDL protocol described in Section 3.6.6 (note that due to the homomorphic nature of Pedersen commitment we don't need to prove that the values in the AL which are leaked to the IP are correct, since both parties can add those to the commitment "in public"). Finally, we use the protocol in Section 3.6.9 for proving that the ciphertext encrypts the right value, and use standard "AND" composition of Σ-protocols to assert that the values appearing in different proofs are consistent.

Implementation of $\Pi_{\mathsf{id-layer}}$. We instantiate the weakly robust PRF scheme (Definition 3.2.2) with Dodis-Yampolskiy PRF [77]. In this case we use ElGamal as the base encryption scheme for the "share-and-encrypt" ad-hoc threshold encryption scheme $\mathbf{TE}$ (Definition 3.2.6). This is because we are encrypting the public key as a group element, which can also be seen as an encryption of the secret key for "ElGamal in the exponent". Note that this allows to both easily prove knowledge of the secret key, and to make sure that the AR's will only learn the AH public key when decrypting. Due to the algebraic nature of the DY PRF, we can efficiently evaluate it inside an MPC protocol as required to implement $\mathcal{F}_{\mathsf{mpc\text{-}prf}}$ using techniques described in [68, 153].

[6]The notions of black-box simulation extractable NIZK and UC-secure NIZK are interchangeable [53, 113].

When creating a new account, we use both SNARKs and Sigma protocols for proving a single composite statement consisting of a circuit-part and an algebraic part using the technique of [9] to obtain SNARK on algebraically committed input. This commitment is used to tie the witness of the Sigma protocol to the witness used in the SNARK. We describe briefly how we use a combination of Σ-protocols and SNARKs in order to prove relation $\mathcal{R}$ in $\Pi_{\mathsf{id-layer}}$. We use the protocol in Section 3.6.10 for proving knowledge of a signature. We use SNARKs on committed input to prove that account holder's attribute list satisfies a certain policy, and for proving that $x \leq \mathsf{Max}_{\mathsf{ACC}}$ for committed x and public $\mathsf{Max}_{\mathsf{ACC}}$. The protocols for proving that $\mathsf{RegID}_{\mathsf{ACC}} = \mathsf{PRF}_{\mathsf{K}}(x)$ and $\mathsf{E}_{\mathsf{ID}} = \mathsf{TEnc}^{\mathsf{n,d}}_{\mathsf{PK}_{\mathsf{AR}}}(\mathsf{IDcred}_{\mathtt{PUB}})$ are described in Section 3.6.10. Note that all the Σ-protocol proofs are made non-interactive using Fiat-Shamir as discussed in Section 3.6.2.

3.8 Transaction Layer

Our paper shows how account holders in our design can create accounts after registering with identity providers. As we have seen, an account includes an account-specific public key where the account holder has the private key. Since our protocol is completely generic, the "key" can in fact be a vector of keys which includes a key for encryption, one for signatures, etc. and therefore accounts can be used for transactions in many different ways.

In this section, we include an informal presentation of one way in which accounts can be used for transferring money on the blockchain, assuming each account has a public signing key and a public encryption key. We stress that this is just as example of one of many possible ways of doing this and, as we prove our ID layer secure in the UC framework one can use our ID layer with any other transaction layer, or even other applications not involving payments.

The system we sketch supports plaintext transactions and encrypted transactions. All transactions must be signed by the account holder from which the transaction originates. When a transaction is published, the ledger will check the signature and allow the transaction to go through if the signature is valid, and possibly if other constraints (described in detail below) are satisfied. Note that payments are linked to the sending and the receiving account and if several payments are made between the same accounts this will be visible on chain. We allow this to have a trade-off between efficiency and privacy: one can make several payments from an account and only suffer the cost of opening it once. On the other hand, one can also choose to use each account once for complete privacy.

Let ACC be an account with encryption key $\mathsf{ek}_{\mathsf{ACC}}$. An account will hold a public amount p and a secret amount s. The secret amount is represented as a set $\mathcal{S} = \{S_i\}_{i=1}^{|\mathcal{S}|}$, where $S_i = \mathsf{Enc}_{\mathsf{ek}_{\mathsf{ACC}}}(s_i)$ and $s = \sum_i s_i$.

Plaintext Transactions.

Plaintext transactions happens by a matched reduction and increment of the public amounts on the sending and receiving accounts.

Encrypted Transactions.

Let $\mathcal{S}_1 = \{S_i\}_{i=1}^{n}$ be the representation of the secret amount for account ACC_1 owned by AH_1, where $S_i = \mathsf{Enc}_{\mathsf{ek}_{\mathsf{ACC}_1}}(s_i)$. To do an encrypted transaction of amount a from ACC_1 to some

account ACC_2, AH_1 proceeds as follows:

1. Compute $s' = \sum_{i=1}^{n} s_i - a$.
2. Create $S' = \mathsf{Enc}_{\mathsf{ek}_{\mathsf{ACC}_1}}(s')$.
3. Create $A = \mathsf{Enc}_{\mathsf{ek}_{\mathsf{ACC}_2}}(a)$.
4. Compute a NIZK proof π that $(S', A, S_1, \ldots, S_n)$ contains numbers $(s', a, s_1, \ldots, s_n)$ such that

$$a \geq 0, \tag{3.1}$$
$$s' \geq 0, \tag{3.2}$$
$$s' = \sum_{i=1}^{n} s_i - a. \tag{3.3}$$

The transaction contains (S', A, π). When the transaction is executed, the ledger checks the proof and if it is valid, then let $\mathcal{S}_1 = \{S'\}$ and $\mathcal{S}_2 = \mathcal{S}_2 \cup \{A\}$, where $\mathcal{S}_2$ is the representation of the secret amount for ACC_2.

Compressing an account.

From the above, it follows that after receiving some number of transactions, the receiving account will contain several encrypted amounts, and this may become impractical to handle at some point. Also, note that an account owner does not actually know the amount on his account until he has decrypted the transactions that come into his account. To solve this, the account owner of ACC_2 can execute a compression transaction, which works as follows:

1. Let $\mathcal{S} = \{S_i\}$ be the representation of existing secret amount for ACC_2. Use decryption key $\mathsf{dk}_{\mathsf{ACC}_2}$ to decrypt all S_i and let s be the sum of the decrypted amounts. Compute $e = \mathsf{Enc}_{\mathsf{ek}_{\mathsf{ACC}_2}}(s)$ and let $\mathcal{S}' = \{e\}$. Compute a NIZK proof π showing that e contains the sum of the amounts in the S_i's.
2. publish a compression transaction that contains the identity of the account and $\mathcal{S}', \pi$.

When a compression transaction appears, the ledger checks the proof π and if it is valid, then update the account of ACC_2 to contain $\mathcal{S}'$ instead of $\mathcal{S}$.

Chapter 4

What Makes Fiat–Shamir zkSNARKs (Updatable SRS) Simulation Extractable?

This chapter presents our results on simulation extractability of Fiat-Shamir zkSNARKs in the updatable setting. The contents of this chapter are taken almost verbatim from [93], where we first presented these results.

4.1 Introduction

Zero-knowledge proof systems, which allow a prover to convince a verifier of an NP statement $\mathcal{R}(\mathsf{x}, \mathsf{w})$ without revealing anything else about the witness w have broad application in cryptography and theory of computation [23, 87, 106]. When restricted to computationally sound proof systems, also called *argument systems*[1], proof size can be shorter than the size of the witness [40]. Zero-knowledge Succinct Non-interactive ARguments of Knowledge (zkSNARKs) are zero-knowledge argument systems that additionally have two succinctness properties: small proof sizes and fast verification. Since their introduction in [135], zk-SNARKs have been a versatile design tool for secure cryptographic protocols. They became particularly relevant for blockchain applications that demand short proofs and fast verification for on-chain storage and processing. Starting with their deployment by Zcash [25], they have seen broad adoption, e.g., for privacy-preserving cryptocurrencies and scalable and private smart contracts in Ethereum.

While research on zkSNARKs has seen rapid progress [24, 26, 35, 99, 115, 116, 129, 130, 141] with many works proposing significant improvements in proof size, verifier and prover efficiency, and complexity of the public setup, less attention has been paid to non-malleable zkSNARKs and succinct signatures of knowledge [42, 59] (sometimes abbreviated SoK or referred to as SNARKY signatures [16, 117]).

Relevance of simulation extractability. Most zkSNARKs are shown only to satisfy a standard knowledge soundness property. Intuitively, this guarantees that a prover that creates a valid proof in isolation knows a valid witness. However, deployments of zkSNARKs in real-world applications, unless they are carefully designed to have application-specific malleability protection, e.g. [25], require a stronger property – *simulation-extractability* (SE) – that corresponds much more closely to existential unforgeability of signatures.

[1] We use both terms interchangeably.

This correspondence is made precise by SoK, which uses an NP-language instance as the public verification key. Instead of signing with the secret key, SoK signing requires knowledge of the NP-witness. Intuitively, an SoK is thus a proof of knowledge (PoK) of a witness that is tied to a message. In fact, many signatures schemes, e.g., Schnorr, can be read as SoK for a specific hard relation, e.g., DL [78]. To model strong existential unforgeability of SoK signatures, even when given an oracle for obtaining signatures on different instances, an attacker must not be able to produce new signatures. Chase and Lysyanskaya [59] model this via the notion of simulation extractability which guarantees extraction of a witness even in the presence of simulated signatures.

In practice, an adversary against a zkSNARK system also has access to proofs computed by honest parties that should be modeled as simulated proofs. The definition of knowledge soundness (KS) ignores the ability of an adversary to see other valid proofs that may occur in real-world applications. For instance, in applications of zkSNARKs in privacy-preserving blockchains, proofs are posted on-chain for all blockchain participants to see. We thus argue that SE is a much more suitable notion for robust protocol design. We also claim that SE has primarily an intellectual cost, as it is harder to prove SE than KS—another analogy here is IND-CCA vs IND-CPA security for encryption. However, we will show that the proof systems we consider are SE out-of-the-box.

Fiat–Shamir-based zkSNARKs. Most modern zkSNARK constructions follow a modular blueprint that involves the design of an information-theoretic interactive protocol, e.g. an Interactive Oracle Proof (IOP) [27], that is then compiled via cryptographic tools to obtain an interactive argument system. This is then turned into a zkSNARK using the Fiat-Shamir transform. By additionally hashing the message, the Fiat-Shamir transform is also a popular technique for constructing signatures. While well-understood for 3-message sigma protocols and justifiable in the ROM [20], Fiat–Shamir should be used with care because there are both counterexamples in theory [107] and real-world attacks in practice when implemented incorrectly [136].

In particular, several schemes such as Sonic [132], Plonk [91], Marlin [62] follow this approach where the information-theoretic object is a multi-message algebraic variant of IOP, and the cryptographic primitive in the compiler is a polynomial commitment scheme (PC) that requires a trusted setup. To date, this blueprint lacks an analysis in the ROM in terms of simulation extractability.

Updatable SRS zkSNARKs. One of the downsides of many efficient zkSNARKs [72, 99, 115, 116, 129, 130, 141] is that they rely on a *trusted setup*, where there is a structured reference string (SRS) that is assumed to be generated by a trusted party. In practice, however, this assumption is not well-founded; if the party that generates the SRS is not honest, they can produce proofs for false statements. If the trusted setup assumption does not hold, knowledge soundness breaks down. Groth et al. [120] propose a setting to tackle this challenge which allows parties – provers and verifiers – to *update* the SRS.[2] The update protocol takes an existing SRS and contributes to its randomness in a verifiable way to obtain a new SRS. The guarantee in this *updatable setting* is that knowledge soundness holds as long as one of the parties updating the SRS is honest. The SRS is also *universal*, in that it does not depend on the relation to be proved but only on an upper

[2]This can be seen as an efficient player-replaceable [104] multi-party computation.

bound on the size of the statement's circuit. Although inefficient, as the SRS size is quadratic in the size of the circuit, [120] set a new paradigm for designing zkSNARKs.

The first universal zkSNARK with updatable and linear size SRS was Sonic proposed by Maller et al. in [132]. Subsequently, Gabizon, Williamson, and Ciobotaru designed Plonk [91] which currently is the most efficient updatable universal zkSNARK. Independently, Chiesa et al. [62] proposed Marlin with comparable efficiency to Plonk.

The challenge of SE in the updatable setting. The notion of simulation-extractability for zkSNARKs which is well motivated in practice, has not been studied in the updatable setting. Consider the following scenario: We assume a "rushing" adversary that starts off with a sequence of updates by malicious parties resulting in a subverted reference string srs. By combining their trapdoor contributions and employing the simulation algorithm, these parties can easily compute a proof to obtain a triple $(\mathsf{srs}, \mathsf{x}, \pi)$ that convinces the verifier of a statement x without knowing a witness. Now, assume that at a later stage, a party produces a triple $(\mathsf{srs}', \mathsf{x}, \pi')$ for the same statement with respect to an updated srs' that has an honest update contribution. We want the guarantee that this party must know a witness corresponding to x. The ability to "maul" the proof π from the old SRS to a proof π' for the new SRS without knowing a witness would clearly violate security. The natural idea is to require that honestly *updated* reference strings are indistinguishable from honestly *generated* reference strings even for parties that previously contributed updates. However, this is not sufficient as the adversary can also rush toward the end of the SRS generation ceremony to perform the last update.

A definition of SE in the updatable setting should take these additional powers of the adversary, which are not captured by existing definitions of SE, into consideration. While generic compilers [3, 127] can be applied to updatable SRS SNARKs to obtain SE, not only do they inevitably incur overheads and lead to efficiency loss, we contend that the standard definition of SE does not suffice in the updatable setting.

4.1.1 Our Contributions

We investigate the non-malleability properties of zkSNARK protocols obtained by FS-compiling multi-message protocols in the updatable SRS setting and give a modular approach to analyze their simulation-extractability. We make the following contributions:

- *Updatable simulation extractability (USE).* We propose a definition of simulation extractability in the updatable SRS setting called USE, that captures the additional power the adversary gets by being able to update the SRS.
- *Theorem for USE of FS-compiled proof systems.* We define three notions in the updatable SRS and ROM, *trapdoor-less zero-knowledge*, a *unique response* property, and *rewinding-based knowledge soundness*. Our main theorem shows that multi-message FS-compiled proof systems that satisfy these notions *are USE out-of-the box*.
- *USE for concrete zkSNARKs.* We prove that the most efficient updatable SRS SNARKS – Plonk/Sonic/Marlin – satisfy the premises of our theorem. We thus show that these zkSNARKs are updatable simulation extractable.
- *SNARKY signatures in the updatable setting.* Our results validate the folklore that the Fiat–Shamir transform is a natural means for constructing signatures of knowledge. This

gives rise to the first SoK in the updatable setting and confirms that a much larger class of zkSNARKs, besides [117], can be lifted to SoK.

- *Broad applicability.* The updatable SRS plus ROM includes both the trusted SRS and the ROM as special cases. This implies the relevance of our theorem for transparent zkSNARKs such as Halo2 and Plonky2 that replace the polynomial commitments of Kate et al. [123] with commitments from Bulletproof [41] and STARKs [28], respectively.

4.1.2 Technical Overview

At a high level, the proof of our main theorem for updatable simulation extractability is along the lines of the simulation extractability proof for FS-compiled sigma protocols from [81]. However, our theorem introduces new notions that are more general to allow us to consider proof systems that are richer than sigma protocols and support an updatable setup. We discuss some of the technical challenges below.

Plonk, Sonic, and Marlin were originally presented as interactive proofs of knowledge that are made non-interactive via the Fiat–Shamir transform. In the following, we denote the underlying interactive protocols by P (for Plonk), S (for Sonic), and M (for Marlin) and the resulting non-interactive proof systems by P_{FS}, S_{FS}, M_{FS} respectively.

Rewinding-Based Knowledge Soundness (RBKS). Following [81], one would have to show that for the protocols we consider, a witness can be extracted from sufficiently many valid transcripts with a common prefix. The standard definition of special soundness for sigma protocols requires the extraction of a witness from any two transcripts with the same first message. However, most zkSNARK protocols do not satisfy this notion. We put forth a notion analogous to special soundness that is more general and applicable to a wider class of protocols. Namely, protocols compiled using multi-round FS that rely on an (updatable) SRS. P, S, and M have more than three messages, and the number of transcripts required for extraction is more than two. Concretely, $(3\mathsf{n}+6)$ for Plonk, $(\mathsf{n}+1)$ for Sonic, and $(2\mathsf{n}+3)$ for Marlin, where n is the number of constraints in the proven circuit. Hence, we do not have a pair of transcripts but a *tree of transcripts*.

Furthermore, the protocols we consider are arguments and rely on a SRS that comes with a trapdoor. An adversary in possession of the trapdoor can produce multiple valid proof transcripts potentially for false statements without knowing any witness. This is true even in the updatable setting, where a trapdoor still exists for any updated SRS. Recall that the standard special soundness definition requires witness extraction from *any* suitably structured tree of accepting transcripts. This means that there are no such trees for false statements.

Instead, we give a rewinding-based knowledge soundness definition with an extractor that proceeds in two steps. It first uses a tree building algorithm $\mathcal{T}$ to obtain a tree of transcripts. In the second step, it uses a tree extraction algorithm $\mathsf{Ext}_{\mathsf{ss}}$ to compute a witness from this tree. Tree-based knowledge soundness guarantees that it is possible to extract a witness from all (but negligibly many) trees of accepting transcripts produced by probabilistic polynomial time (PPT) adversaries. That is, if extraction from such a tree fails, then we break an underlying computational assumption. Moreover, this should hold even against adversaries that contribute to the SRS generation.

Unique Response Protocols (UR). Another property required to show simulation extractability is the unique response property which says that for 3-message sigma protocols, the response of

the prover (3-rd message) is determined by the first message and the challenge [86] (intuitively, the prover can only employ fresh randomness in the first message of the protocol). We cannot use this definition since the protocols we consider have multiple rounds of randomized prover messages. In Plonk, both the first and the third messages are randomized. Although the Sonic prover is deterministic after it picks its first message, the protocol has more than 3 messages. The same holds for Marlin. We propose a generalization of the unique response property called k-UR. It requires that the behavior of the prover be determined by the first k of its messages. For our proof, it is sufficient that Plonk is 3-UR, and Sonic and Marlin are 2-UR.

Trapdoor-Less Zero-Knowledge (TLZK). The premises of our main theorem include two computational properties that do not mention a simulator, RBKS and UR, The theorem states that together with a suitable property for the simulator of the zero-knowledge property, they imply USE. Our key technique is to simulate simulation queries when reducing to RBKS and UR. For this it is convenient that the zero-knowledge simulator be trapdoor-less, that is can produce proofs without relying on the knowledge of the trapdoor. Simulation is based purely on the simulators early control over the challenge. In the ROM this corresponds to a simulator that programs the random oracle and can be understood as a generalization of honest-verifier zero-knowledge for multi-message Fiat–Shamir transformed proof systems with an SRS. We say that such a proof system is k-TLZK, if the simulator only programs the k-th challenge and we construct such simulators for P_{FS}, S_{FS}, and M_{FS}.

Technically we will make use of the k-UR property together with the k-TLZK property to bound the probability that the tree produced by the tree builder $\mathcal{T}$ of RBKS contains any programmed random oracle queries.

4.1.3 Related Work

There are many results on simulation extractability for non-interactive zero-knowledge proofs (NIZKs). First, Groth [114] noticed that a (black-box) SE NIZK is universally-composable (UC) [51]. Then Dodis et al. [78] introduced a notion of (black-box) *true simulation extractability* (i.e., SE with simulation of true statements only) and showed that no NIZK can be UC-secure if it does not have this property.

In the context of zkSNARKs, the first SE zkSNARK was proposed by Groth and Maller [117] and a SE zkSNARK for QAP was designed by Lipmaa [131]. Kosba et al. [127] gave a general transformation from a NIZK to a black-box SE NIZK. Although their transformation works for zkSNARKs as well, the succinctness of the proof system is not preserved by this transformation. Abdolmaleki et al. [3] showed another transformation that obtains non-black-box simulation extractability but also preserves the succinctness of the argument. The zkSNARK of [116] has been shown to be SE by introducing minor modifications to the construction and making stronger assumptions [11, 39]. Recently, [16] showed that the Groth's original proof system from [116] is weakly SE and randomizable. None of these results are for zkSNARKs in the updatable SRS setting or for zkSNARKs obtained via the Fiat–Shamir transformation. The recent work of [94] shows that Fiat–Shamir transformed Bulletproofs are simulation extractable. While they show a general theorem for multi-round protocols, they do not consider a setting with an SRS, and are therefore inapplicable to zkSNARKs in the updatable SRS setting.

4.2 Definitions and Lemmas for Multi-message SRS-based Protocols

Simulation-extractability for multi-message protocols. Most recent SNARK schemes follow the same blueprint of constructing an interactive information-theoretic proof system that is then compiled into a public coin computationally sound scheme using cryptographic tools such as polynomial commitments, and finally made non-interactive via the Fiat–Shamir transformation. Existing results on simulation extractability (for proof systems and signatures of knowledge) for Fiat–Shamir transformed systems work for 3-message protocols without reference string that require two transcripts for standard model extraction, e.g., [81, 146, 149].

In this section, we define properties that are necessary for our analysis of multi-message protocols with a universal updatable SRS. In order to prove simulation-extractability for such protocols, we require more than just two transcripts for extraction. Moreover, in the updatable setting we consider protocols that rely on an SRS where the adversary gets to contribute to the SRS. We first recall the updatable SRS setting and the Fiat-Shamir transform for $(2\mu+1)$-message protocols. Next, we define trapdoor-less zero-knowledge and simulation-extractability which we base on [81] adapted to the updatable SRS setting. Then, to support multi-message SRS-based protocols compiled using the Fiat–Shamir transform, we generalize the unique response property, and define a notion of computational special soundness called rewinding-based knowledge soundness.

Let P and V be PPT algorithms, the former called the *prover* and the latter the *verifier* of a proof system. Both algorithms take a pre-agreed structured reference string srs as input. The structured reference strings we consider are (potentially) updatable, a notion we recall shortly. We focus on proof systems made non-interactive via the multi-message Fiat–Shamir transform presented below where prover and verifier are provided with a random oracle $\mathcal{H}$. We denote by π a proof created by P on input $(\mathsf{srs}, \mathsf{x}, \mathsf{w})$. We say that proof is accepting if $\mathsf{V}(\mathsf{srs}, \mathsf{x}, \pi)$ accepts it.

Let $\mathsf{R}(\mathcal{A})$ denote the set of random tapes of correct length for adversary $\mathcal{A}$ (assuming the given value of security parameter λ), and let $r \leftarrow\!\$\ \mathsf{R}(\mathcal{A})$ denote the random choice of tape r from $\mathsf{R}(\mathcal{A})$.

4.2.1 Updatable SRS Setup Ceremonies

The definition of updatable SRS ceremonies of [120] requires the following algorithms.

- $(\mathsf{srs}, \rho) \leftarrow \mathsf{GenSRS}(\mathcal{R})$ is a PPT algorithm that takes a relation $\mathcal{R}$ and outputs a reference string srs, and correctness proof ρ.
- $(\mathsf{srs}', \rho') \leftarrow \mathsf{Upd}(\mathsf{srs}, \{\rho_j\}_{j=1}^{n})$ is a PPT algorithm that takes a srs, a list of update proofs and outputs an updated srs' together with a proof of correct update ρ'.
- $b \leftarrow \mathsf{VerifySRS}(\mathsf{srs}, \{\rho_j\}_{j=1}^{n})$ takes a reference string srs, a list of update proofs, and outputs a bit indicating acceptance or not.[3]

In the next section, we define security notions in the updatable setting by giving the adversary access to an SRS update oracle UpdO, defined in Fig. 4.1. The oracle allows the adversary

[3]For instance Plonk and Marlin will use the GenSRS, Upd and VerifySRS algorithms in Fig. 4.4.

$\mathsf{UpdO}(\texttt{intent}, \mathsf{srs}_n, \{\rho_j\}_{j=1}^n)$

if $\mathsf{srs} \neq \bot$: **return** $\bot$

if $(\texttt{intent} = \mathsf{Setup})$:

  $(\mathsf{srs}', \rho') \leftarrow \mathsf{GenSRS}(\mathcal{R})$

  $Q_{\mathsf{srs}} \leftarrow Q_{\mathsf{srs}} \cup \{(\mathsf{srs}', \rho')\}$

  return (srs', ρ')

if $(\texttt{intent} = \texttt{update})$:

  $b \leftarrow \mathsf{VerifySRS}(\mathsf{srs}_n, \{\rho_j\}_{j=1}^n)$

  if $(b = 0)$: **return** $\bot$

  $(\mathsf{srs}', \rho') \leftarrow \mathsf{Upd}(\mathsf{srs}_n, \{\rho_j\}_{j=1}^n)$

  $Q_{\mathsf{srs}} \leftarrow Q_{\mathsf{srs}} \cup \{(\mathsf{srs}', \rho')\}$

  return (srs', ρ')

if $(\texttt{intent} = \texttt{final})$:

  $b \leftarrow \mathsf{VerifySRS}(\mathsf{srs}_n, \{\rho_j\}_{j=1}^n)$

  if $(b = 0) \vee Q_{\mathsf{srs}}^{(2)} \cap \{\rho_j\}_i = \varnothing$:

  return $\bot$

    $\mathsf{srs} \leftarrow \mathsf{srs}_n$, **return** srs

else return $\bot$

Figure 4.1: The oracle defines the notion of updatable SRS setup.

to control the SRS generation. A trusted setup can be expressed by the updatable setup definition simply by restricting the adversary to only call the oracle on $\texttt{intent} = \mathsf{Setup}$ and $\texttt{intent} = \texttt{final}$. Note that a soundness adversary now has access to both the random oracle $\mathcal{H}$ and UpdO: $(\mathsf{x}, \pi) \leftarrow \mathcal{A}^{\mathsf{UpdO}, \mathcal{H}}(1^\lambda; r)$.

Remark on universality of the SRS. The proof systems we consider in this work are universal. This means that both the relation $\mathcal{R}$ and the reference string srs allows to prove arithmetic constraints defined over a particular field up to some size bound. The public instance x must determine the constraints. If $\mathcal{R}$ comes with any auxiliary input, the latter is benign. We elide public preprocessing of constraint specific proving and verification keys. While important for performance, this modeling is not critical for security.

4.2.2 Multi-message Fiat-Shamir Compiled Provers and Verifiers

Given interactive prover and (public coin) verifier P', V' that exchange messages resulting in transcript $\tilde{\pi} = (a_1, c_1, \ldots, a_\mu, c_\mu, a_{\mu+1})$, where a_i comes from P' and c_i comes from V', the $(2\mu + 1)$-message Fiat-Shamir heuristic defines non-interactive provers and verifiers P, V as follows:

- P behaves as P' except after sending message a_i, $i \in [1 \,..\, \mu]$, the prover does not wait for the message from the verifier but computes it locally setting $c_i = \mathcal{H}(\tilde{\pi}[0..i])$, where $\tilde{\pi}[0..j] = (\mathsf{x}, a_1, c_1, \ldots, a_{j-1}, c_{j-1}, a_j)$.[4]

[4]For Fiat–Shamir based SoK the message signed m is added to x before hashing.

$\mathsf{SimO}.\mathcal{H}(x)$	$\mathsf{SimO}.\mathsf{Prog}(x,h)$	$\boxed{\mathsf{SimO}.\mathsf{P}(\mathsf{x},\mathsf{w})}$ $\mathsf{SimO}.\mathsf{P}'(\mathsf{x})$
if $H[x] = \bot$ **then**	**if** $H[x] = \bot$ **then**	$\boxed{\textbf{assert } (\mathsf{x},\mathsf{w}) \in \mathcal{R}}$
$\quad H[x] \leftarrow\!\$\ \mathsf{Im}(\mathcal{H})$	$\quad H[x] \leftarrow h$	$\pi \leftarrow \mathsf{Sim}^{\mathsf{SimO}.\mathcal{H},\mathsf{SimO}.\mathsf{Prog}}(\mathsf{srs},\mathsf{x})$
return $H[x]$	$\quad Q_{\mathsf{prog}} \leftarrow Q_{\mathsf{prog}} \cup \{x\}$	$Q \leftarrow Q \cup \{(\mathsf{x},\pi)\}$
	return $H[x]$	**return** π

Figure 4.2: Simulation oracles: srs is the finalized SRS, only $\mathsf{SimO}.\mathsf{P}'$ allows for simulation of false statements

P outputs the non-interactive proof $\pi = (a_1, \ldots, a_\mu, a_{\mu+1})$, that omits challenges as they can be recomputed using $\mathcal{H}$.

- V takes x and π as input and behaves as V' would but does not provide challenges to the prover. Instead it computes the challenges locally as P would, starting from $\tilde{\pi}[0..1] = (\mathsf{x}, a_1)$ which can be obtained from x and π. Then it verifies the resulting transcript $\tilde{\pi}$ as the verifier V' would.

We note that since the verifier can compute the challenges by querying the random oracle, they do not need to be sent by the prover. Thus the π - $\tilde{\pi}$ notational distinction.

Notation for $(2\mu + 1)$-message Fiat–Shamir transformed proof systems. Let $\mathsf{SRS} = (\mathsf{GenSRS}, \mathsf{Upd}, \mathsf{VerifySRS})$ be the algorithm of an updatable SRS ceremony. All our definitions and theorems are about non-interactive proof systems $\boldsymbol{\Psi} = (\mathsf{SRS}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ compiled via the $(2\mu+1)$-message FS transform. That is $\pi = (a_1, \ldots, a_\mu, a_{\mu+1})$ and $\tilde{\pi} = (a_1, c_1, \ldots, a_\mu, c_\mu, a_{\mu+1})$, with $c_i = \mathcal{H}(\tilde{\pi}[0..i])$. We use $\tilde{\pi}[0]$ for instance x and $\tilde{\pi}[i]$, $\tilde{\pi}[i].\mathsf{ch}$ to denote prover message a_i and challenge c_i respectively.

4.2.3 Trapdoor-Less Zero-Knowledge (TLZK)

We call a protocol *trapdoor-less zero-knowledge* (TLZK) if there exists a simulator that does not require the trapdoor, and works by programming the random oracle. Moreover, the simulator may only be allowed to program the random oracle on point $\tilde{\pi}[0, k]$, that is the simulator can only program the challenges that come after the k-th prover message. We call protocols which allow for such a simulation *k-programmable trapdoor-less zero-knowledge*.

Our definition of zero-knowledge for non-interactive arguments is in programmable ROM. We model this using the oracles from Fig. 4.2 that provide a stateful wrapper around Sim. $\mathsf{SimO}.\mathcal{H}(x)$ simulates $\mathcal{H}$ using lazy sampling, $\mathsf{SimO}.\mathsf{Prog}(x, h)$ allows for programming the simulated $\mathcal{H}$ and is available only to Sim. $\mathsf{SimO}.\mathsf{P}(\mathsf{x}, \mathsf{w})$ and $\mathsf{SimO}.\mathsf{P}'(\mathsf{x})$ call the simulator. The former is used in the zero-knowledge definition and requires the statement and witness to be in the relation, the latter is used in the simulation extraction definition and does not require a witness input.

Definition 4.2.1 (Updatable k-Programmable Trapdoor-Less Zero-Knowledge)**.** *Let $\boldsymbol{\Psi}_{\mathsf{FS}} = (\mathsf{SRS}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ be a $(2\mu + 1)$-message FS-transformed NIZK proof system with an updatable SRS setup. We call $\boldsymbol{\Psi}_{\mathsf{FS}}$* trapdoor-less zero-knowledge *with security $\varepsilon_{\mathsf{zk}}$ if for any adversary $\mathcal{A}$,*

$|\varepsilon_0(\lambda) - \varepsilon_1(\lambda)| \leq \varepsilon_{\mathsf{zk}}(\lambda)$, *where*

$$\varepsilon_0(\lambda) = \Pr\left[\mathcal{A}^{\mathsf{UpdO},\mathcal{H},\mathsf{P}}(1^\lambda)\right], \varepsilon_1(\lambda) = \Pr\left[\mathcal{A}^{\mathsf{UpdO},\mathsf{SimO}.\mathcal{H},\mathsf{SimO}.\mathsf{P}}(1^\lambda)\right].$$

If $\varepsilon_{\mathsf{zk}}(\lambda)$ *is negligible, we say* $\boldsymbol{\Psi}_{\mathsf{FS}}$ *is trapdoor-less zero-knowledge. Additionally, we say that* $\boldsymbol{\Psi}_{\mathsf{FS}}$ *is* k*-programmable, if* Sim *before returning a proof* π *only calls* SimO.Prog *on* $(\tilde{\pi}[0..k], h)$*. That is, it only programs the* k*-th message.*

Remark 4.2.2 (TLZK vs HVZK). We note that TLZK notion is closely related to honest-verifier zero-knowledge in the standard model. That is, if we consider an interactive proof system $\boldsymbol{\Psi}$ that is HVZK in the standard model then $\boldsymbol{\Psi}_{\mathsf{FS}}$ is TLZK. This comes as the simulator Sim in $\boldsymbol{\Psi}$ produces a valid simulated proof by picking verifier's challenges according to a predefined distribution and $\boldsymbol{\Psi}_{\mathsf{FS}}$'s simulator $\mathsf{Sim}_{\mathsf{FS}}$ produces its proofs similarly by picking the challenges and additionally programming the random oracle to return the picked challenges. Importantly, in both $\boldsymbol{\Psi}$ and $\boldsymbol{\Psi}_{\mathsf{FS}}$ success of the simulator does not depend on access to an SRS trapdoor.

We note that Plonk is 3-programmable TLZK, and Sonic and Marlin are 2-programmable TLZK. This follows directly from the proofs of their standard model zero-knowledge property in Lemmas 4.6.3,4.7.6,4.8.3.

4.2.4 Updatable Simulation Extractability (USE)

We note that the zero-knowledge property is only guaranteed for statements in the language. For *simulation extractability* where the simulator should be able to provide simulated proofs for false statements as well, we thus use the oracle SimO.P′ [5].

Definition 4.2.3 (Updatable Simulation Extractability)**.** *Let* $\boldsymbol{\Psi}_{\mathsf{NI}} = (\mathsf{SRS}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *be a NIZK proof system with an updatable SRS setup. We say that* $\boldsymbol{\Psi}_{\mathsf{NI}}$ *is* updatable simulation-extractable *with security loss* $\varepsilon_{\mathsf{se}}(\lambda, \mathsf{acc}, q)$ *if for any* PPT *adversary* $\mathcal{A}$ *that is given oracle access to update oracle* UpdO *and simulation oracle* SimO *and that produces an accepting proof for* $\boldsymbol{\Psi}_{\mathsf{NI}}$ *with probability* acc*, where*

$$\mathsf{acc} = \Pr\left[\begin{array}{c|c} \mathsf{V}(\mathsf{srs}, \mathsf{x}, \pi) = 1 & r \leftarrow\!\$\ \mathsf{R}(\mathcal{A}) \\ \wedge (\mathsf{x}, \pi) \notin Q & (\mathsf{x}, \pi) \leftarrow \mathcal{A}^{\mathsf{UpdO},\mathsf{SimO}.\mathcal{H},\mathsf{SimO}.\mathsf{P}'}(1^\lambda; r) \end{array}\right]$$

there exists an expected PPT extractor $\mathsf{Ext}_{\mathsf{se}}$ *such that*

$$\Pr\left[\begin{array}{c|c} \mathsf{V}(\mathsf{srs}, \mathsf{x}, \pi) = 1, & r \leftarrow\!\$\ \mathsf{R}(\mathcal{A}), (\mathsf{x}, \pi) \leftarrow \mathcal{A}^{\mathsf{UpdO},\mathsf{SimO}.\mathcal{H},\mathsf{SimO}.\mathsf{P}'}(1^\lambda; r) \\ (\mathsf{x}, \pi) \notin Q, & \mathsf{w} \leftarrow \mathsf{Ext}_{\mathsf{se}}(\mathsf{srs}, \mathcal{A}, r, Q_{\mathsf{srs}}, Q_{\mathcal{H}}, Q) \\ \mathcal{R}(\mathsf{x}, \mathsf{w}) = 0 & \end{array}\right] \leq \varepsilon_{\mathsf{se}}(\lambda, \mathsf{acc}, q)$$

Here, srs *is the finalized SRS. List* Q_{srs} *contains all* (srs, ρ) *of update SRSs and their proofs, list* $Q_{\mathcal{H}}$ *contains all* $\mathcal{A}$*'s queries to* SimO.$\mathcal{H}$ *and the (simulated) random oracle's answers,* $|Q_{\mathcal{H}}| \leq q$*, and list* Q *contains all* (x, π) *pairs where* x *is an instance queried to* SimO.P′ *by the adversary and* π *is the simulator's answer .*

[5] Note, that simulation extractability property where the simulator is required to give simulated proofs for true statements only is called *true simulation extractability.*

4.2.5 Unique Response (UR) Protocols

A technical hurdle identified by Faust et al. [81] for proving simulation extraction via the Fiat–Shamir transformation is that the transformed proof system satisfies a unique response property. The original formulation by Fischlin, although suitable for applications presented in [81, 86], does not suffice in our case. First, the property assumes that the protocol has three messages, with the second being the challenge from the verifier. That is not the case we consider here. Second, it is not entirely clear how to generalize the property. Should one require that after the first challenge from the verifier, the prover's responses are fixed? That does not work since the prover needs to answer differently on different verifier's challenges, as otherwise the protocol could have fewer messages. Another problem is that the protocol could have a message, beyond the first prover's message, which is randomized. Unique response cannot hold in this case. Finally, the protocols we consider here are not in the standard model, but use an SRS.

We work around these obstacles by providing a generalized notion of the unique response property. More precisely, we say that a $(2\mu+1)$-message protocol has *unique responses from* k, and call it a k-UR-protocol, if it follows the definition below:

Definition 4.2.4 (Updatable k-Unique Response Protocol)**.** *Let* $\mathbf{\Psi}_{\mathsf{FS}} = (\mathsf{SRS}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *be a* $(2\mu+1)$*-message FS-transformed NIZK proof system with an updatable SRS setup. Let* $\mathcal{H}$ *be the random oracle. We say that* $\mathbf{\Psi}_{\mathsf{FS}}$ *has* unique responses for k *with security* $\varepsilon_{\mathsf{ur}}(\lambda)$ *if for any* PPT *adversary* $\mathcal{A}_{\mathsf{ur}}$*:*

$$\Pr\left[\begin{array}{l|l} \pi \neq \pi', \tilde{\pi}[0..k] = \tilde{\pi}'[0..k], & \\ \mathsf{V}'(\mathsf{srs}, \mathsf{x}, \pi, c) = \mathsf{V}'(\mathsf{srs}, \mathsf{x}, \pi', c) = 1 & (\mathsf{x}, \pi, \pi', c) \leftarrow \mathcal{A}_{\mathsf{ur}}^{\mathsf{UpdO}, \mathcal{H}}(1^\lambda) \end{array}\right] \leq \varepsilon_{\mathsf{ur}}(\lambda)$$

where srs *is the finalized SRS and* $\mathsf{V}'(\mathsf{srs}, \mathsf{x}, \pi = (a_1, \ldots, a_\mu, a_{\mu+1}), c)$ *behaves as* $\mathsf{V}(\mathsf{srs}, \mathsf{x}, \pi)$ *except for using* c *as the* k*-th challenge instead of calling* $\mathcal{H}(\tilde{\pi}[0..k])$*. Thus,* $\mathcal{A}$ *can program the* k*-th challenge. We say* $\mathbf{\Psi}_{\mathsf{FS}}$ *is* k-UR*, if* $\varepsilon_{\mathsf{ur}}(\lambda)$ *is negligible.*

Intuitively, a protocol is k-UR if it is infeasible for a PPT adversary to produce a pair of accepting proofs $\pi \neq \pi'$ that are the same on the first k messages of the prover.

The definition can be easily generalized to allow for programing the oracle on more than just a single point. We opted for this simplified presentation, since all the protocols analyzed in this paper require only single-point programming,

4.2.6 Rewinding-Based Knowledge Soundness (RBKS)

Before giving the definition of rewinding-based knowledge soundness for NIZK proof systems compiled via the $(2\mu+1)$-message FS transformation, we first recall the notion of a tree of transcripts.

Definition 4.2.5 (Tree of accepting transcripts, cf. [38])**.** *A* $(n_1, \ldots, n_\mu)$*-tree of accepting transcripts is a tree where each node on depth* i*, for* $i \in [1 \,..\, \mu+1]$*, is an* i*-th prover's message in an accepting transcript; edges between the nodes are labeled with challenges, such that no two edges on the same depth have the same label; and each node on depth* i *has* $n_i - 1$ *siblings and* n_{i+1} *children. The tree consists of* $N = \prod_{i=1}^{\mu} n_i$ *branches, where* N *is the number of accepting transcripts. We require* $N = \mathsf{poly}(\lambda)$*. We refer to a* $(1, \ldots, n_k = n, 1, \ldots, 1)$*-tree as a* (k, n)*-tree.*

The existence of simulation trapdoor for $\mathbf{P}$, $\mathbf{S}$ and $\mathbf{M}$ means that they are not special sound in the standard sense. We therefore put forth the notion of rewinding-based knowledge soundness that is a computational notion. Note that in the definition below, it is implicit that each transcript in the tree is accepting with respect to a "local programming" of the random oracle. However, the verification of the proof output by the adversary is with respect to a non-programmed random oracle.

Definition 4.2.6 (Updatable Rewinding-Based Knowledge Soundness)**.** *Let $n_1, \ldots, n_\mu \in \mathbb{N}$. Let $\mathbf{\Psi}_{\mathsf{FS}} = (\mathsf{SRS}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ be a $(2\mu+1)$-message FS-transformed NIZK proof system with an updatable SRS setup for relation $\mathcal{R}$. Let $\mathcal{H}$ be the random oracle. We require existence of an expected PPT tree builder $\mathcal{T}$ that eventually outputs a T which is either a $(n_1, \ldots, n_\mu)$-tree of accepting transcript or $\perp$ and a PPT extractor $\mathsf{Ext}_{\mathsf{ss}}$. Let adversary $\mathcal{A}_{\mathsf{ks}}$ be a PPT algorithm, that outputs a valid proof with probability at least acc, where*

$$\mathsf{acc} = \Pr\left[\begin{array}{c|c} \mathsf{V}(\mathsf{srs}, \mathsf{x}, \pi) = 1 & r \leftarrow\!\!\$\ \mathsf{R}(\mathcal{A}_{\mathsf{ks}}) \\ \wedge\ (\mathsf{x}, \pi) \notin Q & (\mathsf{x}, \pi) \leftarrow \mathcal{A}_{\mathsf{ks}}^{\mathsf{UpdO}, \mathcal{H}}(1^\lambda; r) \end{array}\right].$$

We say that $\mathbf{\Psi}_{\mathsf{FS}}$ is $(n_1, \ldots, n_\mu)$-rewinding-based knowledge sound with security loss $\varepsilon_{\mathsf{ks}}(\lambda, \mathsf{acc}, q)$ if

$$\Pr\left[\begin{array}{c|l} \mathsf{V}(\mathsf{srs}, \mathsf{x}, \pi) = 1, & r \leftarrow\!\!\$\ \mathsf{R}(\mathcal{A}_{\mathsf{ks}}), \\ \wedge\ \mathcal{R}(\mathsf{x}, \mathsf{w}) = 0 & (\mathsf{x}, \cdot) \leftarrow \mathcal{A}_{\mathsf{ks}}^{\mathsf{UpdO}, \mathcal{H}}(1^\lambda; r) \\ & \mathsf{T} \leftarrow \mathcal{T}(\mathsf{srs}, \mathcal{A}_{\mathsf{ks}}, r, Q_{\mathsf{srs}}, Q_{\mathcal{H}}), \mathsf{w} \leftarrow \mathsf{Ext}_{\mathsf{ss}}(\mathsf{T}) \end{array}\right] \leq \varepsilon_{\mathsf{ks}}(\lambda, \mathsf{acc}, q).$$

Here, srs is the finalized SRS. List Q_{srs} contains all (srs, ρ) of updated SRSs and their proofs, and list $Q_{\mathcal{H}}$ contains all of the adversaries queries to $\mathcal{H}$ and the random oracle's answers, $|Q_{\mathcal{H}}| \leq q$. We call the protocol (k, n)-rewinding-based knowledge sound if T is a (k, n)-tree of accepting transcripts.

4.3 Simulation Extractability—The General Result

Equipped with the definitional framework of Section 4.2, we now present the main result of this paper: a proof of simulation extractability for multi-message Fiat–Shamir-transformed NIZK proof systems.

Without loss of generality, we assume that whenever the accepting proof contains a response to a challenge from a random oracle, then the adversary queried the oracle to get it. It is straightforward to transform any adversary that violates this condition into an adversary that makes these additional queries to the random oracle and wins with the same probability.

The core conceptual insight of the proof is that the k-unique response and k-programmable trapdoor-less zero-knowledge properties together ensure that the k-th move challenges in the trees of rewinding-based knowledge soundness are fresh and do not come from the simulator. This allows us to eliminate the simulation oracle in our rewinding argument and enables us to use the existing results of [12] in later sections.

Theorem 4.3.1 (Simulation-extractable multi-message protocols)**.** *Let $\mathbf{\Psi}_{\mathsf{FS}} = (\mathsf{SRS}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ be a $(2\mu+1)$-message FS-transformed NIZK proof system with an updatable SRS setup. If $\mathbf{\Psi}_{\mathsf{FS}}$ is an updatable k-unique response protocol with security loss $\varepsilon_{\mathsf{ur}}$, is updatable k-programmable*

$\mathsf{SimO}.\mathcal{H}(x)$	$\mathsf{SimO}.\mathsf{Prog}(x,h)$
if $H[x]=\bot$ **then**	**if** $H[x]=\bot$ **then**
$\quad H[x]\leftarrow\mathcal{H}(x)$	$\quad H[x]\leftarrow h$
return $H[x]$	$\quad Q_{\mathsf{prog}}\leftarrow Q_{\mathsf{prog}}\cup\{x\}$
	return $H[x]$

Figure 4.3: Simulating random oracle calls.

trapdoor-less zero-knowledge, and $(1,\ldots,1,n_k,\ldots,n_\mu)$*-updatable rewinding-based knowledge sound (for some* $n_k,\ldots,n_\mu$*) with security loss* $\varepsilon_{\mathsf{ks}}$*; Then* $\mathbf{\Psi}_{\mathsf{FS}}$ *is* updatable simulation-extractable *with security loss*

$$\varepsilon_{\mathsf{se}}(\lambda,\mathsf{acc},q)\leq\varepsilon_{\mathsf{ks}}(\lambda,\mathsf{acc}-\varepsilon_{\mathsf{ur}}(\lambda),q)$$

against any PPT *adversary* $\mathcal{A}$ *that makes up to* q *random oracle queries and returns an accepting proof with probability at least* acc*.*

Proof. Let $(\mathsf{x},\pi)\leftarrow\mathcal{A}^{\mathsf{UpdO},\mathsf{SimO}.\mathcal{H},\mathsf{SimO}.\mathsf{P}'}(r_{\mathcal{A}})$ be the USE adversary. We show how to build an extractor $\mathsf{Ext}_{\mathsf{se}}(\mathsf{srs},\mathcal{A},r_{\mathcal{A}},Q,Q_{\mathcal{H}},Q_{\mathsf{srs}})$ that outputs a witness w, such that $\mathcal{R}(\mathsf{x},\mathsf{w})$ holds with high probability. To that end we define an algorithm $\mathcal{A}_{\mathsf{ks}}^{\mathsf{UpdO},\mathcal{H}}(r)$ against rewinding-based knowledge soundness of $\mathbf{\Psi}_{\mathsf{FS}}$ that runs internally $\mathcal{A}^{\mathsf{UpdO},\mathsf{SimO}.\mathcal{H},\mathsf{SimO}.\mathsf{P}'}(r_{\mathcal{A}})$. Here $r=(r_{\mathsf{Sim}},r_{\mathcal{A}})$ with r_{Sim} the randomness that will be used to simulate $\mathsf{SimO}.\mathsf{P}'$.

The code of $\mathcal{A}_{\mathsf{ks}}^{\mathsf{UpdO},\mathcal{H}}(r)$ hardcodes Q such that it does not use any randomness for proofs in Q as long as statements are queried in order. In this case it simple returns a proof π_{Sim} from Q but nevertheless queries $\mathsf{SimO}.\mathsf{Prog}$ on $(\tilde{\pi}_{\mathsf{Sim}}[0..k],\tilde{\pi}_{\mathsf{Sim}}[k].ch)$, i.e. it programs the k-th challenge. While it is hard to construct such an adversary without knowing Q, it clearly exists and $\mathsf{Ext}_{\mathsf{se}}$ has the necessary inputs to construct $\mathcal{A}_{\mathsf{ks}}$. This hardcoding guarantees that $\mathcal{A}_{\mathsf{ks}}$ returns the same (x,π) as $\mathcal{A}$ in the experiment. Eventually, $\mathsf{Ext}_{\mathsf{se}}$ uses the tree builder $\mathcal{T}$ and extractor $\mathsf{Ext}_{\mathsf{ss}}$ for $\mathcal{A}_{\mathsf{ks}}$ to extract the witness for x. Both guaranteed to exist (and be successful with high probability) by rewinding-based knowledge soundness. This high-level argument shows that $\mathsf{Ext}_{\mathsf{se}}$ exists as well.

We now give the details of the simulation that guarantees that $\mathcal{A}_{\mathsf{ks}}$ is successful whenever $\mathcal{A}$ is—except with a small security loss that we will bound late: Since $\mathcal{A}_{\mathsf{ks}}$ runs $\mathcal{A}$ internally, it needs to take care of $\mathcal{A}$'s oracle queries. $\mathcal{A}_{\mathsf{ks}}$ passes on queries of $\mathcal{A}$ to the update oracle UpdO to its own UpdO oracle and returns the result to $\mathcal{A}$. $\mathcal{A}_{\mathsf{ks}}$ internally simulates (non-hardcoded) queries to the simulator $\mathsf{SimO}.\mathsf{P}'$ by running the Sim algorithm on randomness r_{Sim} of its tape. Sim requires access to oracles $\mathsf{SimO}.\mathcal{H}$ to compute a challenge honestly and $\mathsf{SimO}.\mathsf{Prog}$ to program a challenge. Again $\mathcal{A}_{\mathsf{ks}}$ simulates both of these oracles internally, cf. Fig. 4.3, this time using the $\mathcal{H}$ oracle of $\mathcal{A}_{\mathsf{ks}}$. Note that queries of $\mathcal{A}$ to $\mathsf{SimO}.\mathcal{H}$ are not programmed, but passed on to $\mathcal{H}$.

Importantly, all challenges in simulated proofs, up to round k are also computed honestly, i.e. $\tilde{\pi}[i].\mathsf{ch}=\mathcal{H}(\tilde{\pi}[0..i])$, for $i<k$.

Eventually, $\mathcal{A}$ outputs an instance and proof (x,π). $\mathcal{A}_{\mathsf{ks}}$ returns the same values as long as $\tilde{\pi}[0..i]\notin Q_{\mathsf{prog}}$, $i\in[1,\mu]$. This models that the proof output by $\mathcal{A}_{\mathsf{ks}}$ must not contain any programmed queries as such a proof would not be w.r.t $\mathcal{H}$ in the RBKS experiment. If $\mathcal{A}$ outputs a proof that does contain programmed challenges, then $\mathcal{A}_{\mathsf{ks}}$ aborts. We denote this event by E.

Lemma 4.3.2. *Probability that* E *happens is upper-bounded by* $\varepsilon_{\mathsf{ur}}(\lambda)$*.*

Proof. We build an adversary $\mathcal{A}_{\mathsf{ur}}^{\mathsf{UpdO},\mathcal{H}}(\lambda; r)$ that has access to the random oracle $\mathcal{H}$ and update oracle UpdO. $\mathcal{A}_{\mathsf{ur}}$ uses $\mathcal{A}_{\mathsf{ks}}$ to break the k-UR property of $\boldsymbol{\Psi}_{\mathsf{FS}}$.

When $\mathcal{A}_{\mathsf{ks}}$ outputs a proof π for x such that E holds, $\mathcal{A}_{\mathsf{ur}}$ looks through lists Q and $Q_{\mathcal{H}}$ until it finds $\tilde{\pi}_{\mathsf{Sim}}[0..k]$ such that $\tilde{\pi}[0..k] = \tilde{\pi}_{\mathsf{Sim}}[0..k]$ and a programmed random oracle query $\tilde{\pi}_{\mathsf{Sim}}[k].\mathsf{ch}$ on $\tilde{\pi}_{\mathsf{Sim}}[0..k]$. $\mathcal{A}_{\mathsf{ur}}$ returns two proofs π and π_{Sim} for x:

$$\begin{aligned}\pi_1 = \pi_{\mathsf{Sim}} &= (\pi_{\mathsf{Sim}}[1..k], \pi_{\mathsf{Sim}}[k+1..\mu+1])\\ \pi_2 = \pi \quad &= (\pi_{\mathsf{Sim}}[1..k], \quad \pi[k+1..\mu+1])\end{aligned}$$

and the challenge $\tilde{\pi}_{\mathsf{Sim}}[k].\mathsf{ch} = \tilde{\pi}[k].\mathsf{ch}$.

Importantly, both proofs are w.r.t the unique response verifier. The first, since it is a correctly computed simulated proof for which the unique response property definition allows any challenges at k. The latter, since it is an accepting proof produced by the adversary. We have that $\pi \neq \pi_{\mathsf{Sim}}$ as otherwise $\mathcal{A}$ does not win the simulation extractability game as $\pi \in Q$. On the other hand, if the proofs are different, then $\mathcal{A}_{\mathsf{ur}}$ breaks k-UR-ness of $\boldsymbol{\Psi}_{\mathsf{FS}}$. This happens only with probability $\varepsilon_{\mathsf{ur}}(\lambda)$. □

We denote by $\widetilde{\mathsf{acc}}$ the probability that $\mathcal{A}_{\mathsf{ks}}$ outputs an accepting proof. We note that by up-to-bad reasoning $\widetilde{\mathsf{acc}}$ is at most $\varepsilon_{\mathsf{ur}}(\lambda)$ far from the probability that $\mathcal{A}$ outputs a proof. Thus, the probability that $\mathcal{A}_{\mathsf{ks}}$ outputs a proof is at least $\widetilde{\mathsf{acc}} \geq \mathsf{acc} - \varepsilon_{\mathsf{ur}}(\lambda)$. Since $\boldsymbol{\Psi}_{\mathsf{FS}}$ is $\varepsilon_{\mathsf{ks}}(\lambda, \widetilde{\mathsf{acc}}, q)$ rewinding-based knowledge sound, there is a tree builder $\mathcal{T}$ and extractor $\mathsf{Ext}_{\mathsf{ss}}$ that rewinds $\mathcal{A}_{\mathsf{ks}}$ to obtain a tree of accepting transcripts T and fails to extract the witness with probability at most $\varepsilon_{\mathsf{ks}}(\lambda, \widetilde{\mathsf{acc}}, q)$. The extractor $\mathsf{Ext}_{\mathsf{se}}$ outputs the witness with the same probability.

Thus $\varepsilon_{\mathsf{se}}(\lambda, \mathsf{acc}, q) = \varepsilon_{\mathsf{ks}}(\lambda, \widetilde{\mathsf{acc}}, q) \leq \varepsilon_{\mathsf{ks}}(\lambda, \mathsf{acc} - \varepsilon_{\mathsf{ur}}, q)$. □

Remark 4.3.3. Observe that our theorem does not depend on $\varepsilon_{\mathsf{zk}}(\lambda)$. There is no real prover algorithm P in the experiment. Only the k-programmability of TLZK matters.

Remark 4.3.4. Observe that the theorem does not prescribe a tree shape for the tree builder $\mathcal{T}$. Interestingly, in our concrete results $\mathcal{T}$ outputs a $(k, *)$-tree of accepting transcripts.

4.4 Polynomial Commitment Schemes

A polynomial commitment scheme $\mathbf{PC} = (\mathsf{GenSRS}, \mathsf{Com}, \mathsf{Op}, \mathsf{Verify})$ consists of four algorithms and allows to commit to a polynomial f and later open the evaluation in a point z to some value $s = \mathsf{f}(z)$. More formally:

$\mathsf{GenSRS}(1^\lambda, \mathsf{max})$**:** The key generation algorithm takes in a security parameter λ and a parameter max which determines the maximal degree of the committed polynomial. It outputs a structured reference string srs (the commitment key). Note that srs implicitly determines λ and max.

$\mathsf{Com}(\mathsf{srs}, \mathsf{f})$**:** The commitment algorithm $\mathsf{Com}(\mathsf{srs}, \mathsf{f})$ takes in srs and a polynomial f with maximum degree max, and outputs a commitment c.

$\mathsf{Op}(\mathsf{srs}, z, s, \mathsf{f})$**:** The opening algorithm takes as input srs, an evaluation point z, a value s and the polynomial f. It outputs an opening o.

$\mathsf{Verify}(\mathsf{srs}, c, z, s, o)$**:** The verification algorithm takes in srs, a commitment c, an evaluation point z, a value s and an opening o. It outputs 1 if o is a valid opening for (c, z, s) and 0 otherwise.

A secure polynomial commitment $\mathbf{PC}$ should satisfy correctness, evaluation binding, opening uniqueness, hiding and knowledge-binding. Note that since we are in the updatable setting, srs in these security definitions is the SRS that $\mathcal{A}$ finalizes using the update oracle UpdO (See Fig. 4.1).

Evaluation binding: A PPT adversary $\mathcal{A}$ which outputs a commitment $\boldsymbol{c}$ and evaluation points $\boldsymbol{z}$ has at most negligible chances to open the commitment to two different evaluations $\boldsymbol{s}, \boldsymbol{s'}$. That is, let $k \in \mathbb{N}$ be the number of committed polynomials, $l \in \mathbb{N}$ number of evaluation points, $\boldsymbol{c} \in \mathbb{G}^k$ be the commitments, $\boldsymbol{z} \in \mathbb{F}_p^l$ be the arguments the polynomials are evaluated at, $\boldsymbol{s}, \boldsymbol{s'} \in \mathbb{F}_p^k$ the evaluations, and $\boldsymbol{o}, \boldsymbol{o'} \in \mathbb{F}_p^l$ be the commitment openings. Then for every PPT adversary $\mathcal{A}$

$$\Pr\left[\begin{array}{c}\mathsf{Verify}(\mathsf{srs}, \boldsymbol{c}, \boldsymbol{z}, \boldsymbol{s}, \boldsymbol{o}) = 1, \\ \mathsf{Verify}(\mathsf{srs}, \boldsymbol{c}, \boldsymbol{z}, \boldsymbol{s'}, \boldsymbol{o'}) = 1, \\ \boldsymbol{s} \neq \boldsymbol{s'}\end{array} \middle| \; (\boldsymbol{c}, \boldsymbol{z}, \boldsymbol{s}, \boldsymbol{s'}, \boldsymbol{o}, \boldsymbol{o'}) \leftarrow \mathcal{A}^{\mathsf{UpdO}}(\mathsf{max}) \right] \leq \mathsf{negl}(\lambda)\,.$$

Hiding: We also formalize notion of k-hiding property of a polynomial commitment scheme. Let H be a set of size $\mathsf{max}+1$ and $\mathsf{Z_H}$ its vanishing polynomial. We say that a polynomial scheme is *hiding* with security $\varepsilon_{\mathsf{hid}}(\lambda)$ if for every PPT adversary $\mathcal{A}$, $k \in \mathbb{N}$, probability

$$\Pr\big[b' = b \;\big|\; (f_0, f_1, c, k, b') \leftarrow \mathcal{A}^{\mathsf{UpdO}, \mathsf{O}_C}(\mathsf{max}, c), f_0, f_1 \in \mathbb{F}^{\mathsf{max}}[X]\big] \leq \frac{1}{2} + \varepsilon(\lambda).$$

Where $c = f'_b(\chi)$, for a random bit b and the polynomial $f'_b(X) = f_b + \mathsf{Z_H}(X)(a_0 + a_1 X + \ldots a_{k-1}X^{k-1})$, and the oracle O_C on adversary's evaluation query x it adds x to initially empty set Q_x and if $|Q_x| \leq k$, it provides $f'_b(x)$.

Commitment of knowledge: Intuitively, when a commitment scheme is "of knowledge" then if an adversary produces a (valid) commitment c, which it can open correctly in an evaluation point, then it must know the underlying polynomial f which commits to that value. For every PPT adversary $\mathcal{A}$ who produces commitment c, evaluation s and opening o there exists a PPT extractor Ext such that

$$\Pr\left[\begin{array}{c}\deg \mathsf{f} \leq \mathsf{max}, c = \mathsf{Com}(\mathsf{srs}, \mathsf{f}), \\ \mathsf{Verify}(\mathsf{srs}, c, z, s, o) = 1\end{array} \middle| \begin{array}{c} c \leftarrow \mathcal{A}^{\mathsf{UpdO}}(\mathsf{max}), z \leftarrow\!\!\$\, \mathbb{F}_p \\ (s, o) \leftarrow \mathcal{A}(c, z), \\ \mathsf{f} = \mathsf{Ext}_{\mathcal{A}}(\mathsf{srs}, c)\end{array} \right] \geq 1 - \varepsilon_{\mathsf{k}}(\lambda).$$

In that case we say that $\mathbf{PC}$ is $\varepsilon_{\mathsf{k}}(\lambda)$-knowledge.

4.4.1 Unique Opening Property of $\mathsf{PC_P}$

Now, we show that the batched variant of the KZG polynomial commitment scheme that is used in Plonk and Marlin, has the unique opening property.

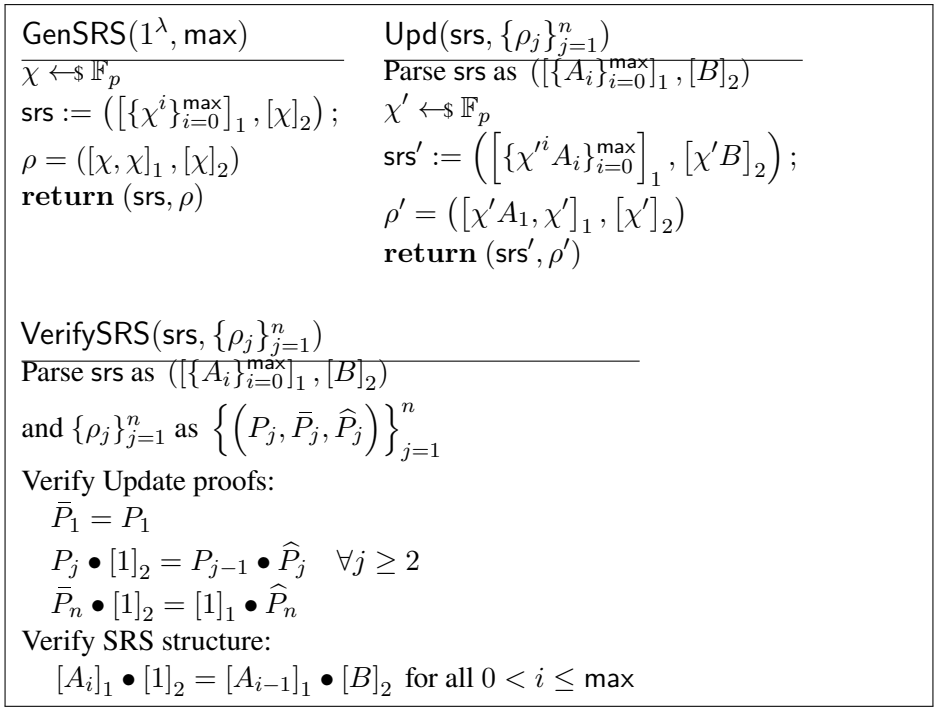

Figure 4.4: Updatable SRS scheme SRS for $\mathbf{PC_P}$

Lemma 4.4.1. *Let* $\mathbf{PC_P} = (\mathsf{GenSRS}, \mathsf{Com}, \mathsf{Op}, \mathsf{VerifyBatched})$ *be a batched version of a KZG polynomial commitment, cf. Fig. 4.5, that commits to* m *polynomials of degree up to* max*. Let* $\boldsymbol{z} = (z_0, \ldots, z_{l-1}) \in \mathbb{F}_p^l$ *be the points the polynomials are evaluated at,* $k_i \in \mathbb{N}$ *be the number of the committed polynomials to be evaluated at* z_i*, and* K_i *be the set of indices of these polynomials. Let* $\boldsymbol{s}_{\boldsymbol{K_i}} \in \mathbb{F}_p^{k_i}$ *be the evaluations of polynomials at* z_i*, and* $\boldsymbol{o} = (o_0, \ldots, o_{l-1}) \in \mathbb{F}_p^l$ *be the commitment openings. We show that the probability an algebraic adversary* $\mathcal{A}$*, who can made up to* q *random oracle queries, opens the same vector of commitments in two different ways is at most* $\varepsilon_{\mathsf{op}}(\lambda)$*, for* $\varepsilon_{\mathsf{op}}(\lambda) \leq l \cdot \varepsilon_{\mathsf{udlog}}(\lambda) + q/p$*, where* $\varepsilon_{\mathsf{udlog}}(\lambda)$ *is security of the* $(\mathsf{max}, 1)$-udlog *assumption and* p *is the field size used in* $\mathbf{PC_P}$*.*

Proof. The proof goes by game hops. In the first game the adversary wins if it presents two accepting openings of a vector of polynomials. Then, we restrict the winning condition and require that the adversary also makes the idealized batched verifier to accept the proof. In the next game, we abort if the idealized verifier rejects a proof for one of the evaluation point.

Game 0: In this game the adversary wins if it provides two different openings for a vector of polynomial commitments and their evaluations that are accepting by VerifyBatched.

Game 1: This game is identical to Game 0 except it is additionally aborted if the commitment opening are not accepting by VerifyBatched′.

Game 0 to Game 1: Any discrepancy between the idealized verifier rejection and real verifier acceptance allows one to break the (updatable) discrete logarithm problem. The reduction $\mathcal{R}_{\mathsf{udlog}}$ proceeds as follows. It answers $\mathcal{A}$'s queries for SRS updates according to the answers it receives from its udlog update oracle. When $\mathcal{A}$ finalizes an SRS, $\mathcal{R}_{\mathsf{udlog}}$ finalizes the corresponding udlog challenge $([1, \ldots, \chi'^{\mathsf{max}}]_1, [1]_2)$. We consider verification equation $(**)$ as a polynomial in X and the verification equation $(*)$ as it's evaluation at χ'. Consider an opening such that

$\mathsf{SRS}(1^\lambda, \mathsf{max})$

cf. Fig. 4.4

$\mathsf{Com}(\mathsf{srs}, \mathbf{f}(X))$

return $[\boldsymbol{c}]_1 = [\mathbf{f}(\chi)]_1$

return $\mathbf{f}(X)$

$\mathsf{Op}(\mathsf{srs}, \boldsymbol{z}, \boldsymbol{s}, \mathbf{f}(X), \mathsf{aux}_0)$

$\boldsymbol{\gamma} \leftarrow \mathcal{H}(g_0(\boldsymbol{z}, \boldsymbol{s}, [\boldsymbol{c}]_1, \mathsf{aux}_0))$

for $i \in [1 \,..\, |\boldsymbol{z}|]$ **do**

$$\mathsf{o}_j(X) \leftarrow \sum_{i \in K_j} \gamma_j^{i-1} \frac{\mathsf{f}_i(X) - \mathsf{f}_i(z_j)}{X - z_j}$$

return $\boldsymbol{o} = [\mathbf{o}(\chi)]_1$

return $\mathbf{o}(X)$

$\mathsf{VerifyBatched}(\mathsf{srs}, [\boldsymbol{c}]_1, \boldsymbol{z}, \boldsymbol{s}, [\mathsf{o}(\chi)]_1, (\mathsf{aux}_0, \mathsf{aux}_1))$

$\boldsymbol{\gamma} \leftarrow \mathcal{H}(g_0(\boldsymbol{z}, \boldsymbol{s}, [\boldsymbol{c}]_1, \mathsf{aux}_0))$

$r \leftarrow \mathcal{H}(g_1([\boldsymbol{c}]_1, \boldsymbol{z}, \boldsymbol{s}, [\mathsf{o}(\chi)]_1, \mathsf{aux}_1))$

$(*)$**if** $\sum_{j=1}^{|\boldsymbol{z}|} r^j \cdot \left[\sum_{i \in K_j} \gamma_j^{i-1} c_i - \sum_{i \in K_j} \gamma_j^{i-1} s_{i_j}\right]_1 \bullet [1]_2 +$
$\sum_{j=1}^{|\boldsymbol{z}|} r^j z_j o_j \bullet [1]_2 \neq \left[\sum_{j=1}^{|\boldsymbol{z}|} r^j o_j\right]_1 \bullet [\chi]_2$ **then**

return 0

$(**)$ **if** $\sum_{j=1}^{|\boldsymbol{z}|} r^j \cdot (\sum_{i \in K_j} \gamma_j^{i-1} \mathsf{f}_i(X) - \sum_{i \in K_j} \gamma_j^{i-1} s_{i_j}) +$
$\sum_{j=1}^{|\boldsymbol{z}|} r^j z_j \mathsf{o}_j(X) \neq \sum_{j=1}^{|\boldsymbol{z}|} r^j \mathsf{o}_j(X) \cdot X$ **then**
return 0

return 1.

$\mathsf{Verify}(\mathsf{srs}, [\boldsymbol{c}]_1, \boldsymbol{z}, \boldsymbol{s}, [\mathsf{o}(\chi)]_1, \mathsf{aux}_0)$

$\boldsymbol{\gamma} \leftarrow \mathcal{H}(g_0(\boldsymbol{z}, \boldsymbol{s}, [\boldsymbol{c}]_1, \mathsf{aux}_0))$

for $j \in [1 \,..\, |\boldsymbol{z}|]$ **do**

if $\left[\sum_{i \in K_j} \gamma_j^{i-1} c_i - \sum_{i \in K_j} \gamma_j^{i-i} s_{i,j}\right]_1 \bullet [1]_2 +$
$z_j o_j \bullet [1]_2 \neq [o_j]_1 \bullet [\chi]_2$ **then**

return 0

if $\sum_{i \in K_j} \gamma_j^{i-1} \mathsf{f}_i(X) - \sum_{i \in K_j} \gamma_j^{i-i} s_{i,j} +$
$z_j \mathsf{o}_j(X) \neq$
$\mathsf{o}_j(X) X$ **then return** 0

return 1.

Figure 4.5: $\mathbf{PC_P}$ polynomial commitment scheme. Here $|\boldsymbol{z}| = l$ is the number of evaluation points, the number of committed polynomials is m, K_j is the set of polynomials that was evaluated at point z_j. Functions g_0 and g_1 are injective and specific to the context in which the polynomial commitment is used. (In our case, functions g_0 and g_1 are produce partial transcripts of the proof that utilizes the commitment scheme, aux contains all additional information that is needed by the functions.) In the boxes we describe values returned or equality computed in the ideal protocol where the verifier checks equalities on the polynomials instead of their evaluations. For algorithm Alg we denote its ideal variant by Alg'.

verification equation (**), cf. Fig. 4.5, does not hold, i.e. (**) is not a zero polynomial, but (*) does, i.e. (**) zeroes at χ'. Since $\mathcal{A}$ is algebraic, all proof elements are extended by their representation as a combination of the input $\mathbb{G}_1$-elements. Therefore, all coefficients of the verification equation polynomial related to (**) are known. Now, $\mathcal{R}_{\mathsf{udlog}}$ computes the roots of (**), finds χ' among them, and returns χ'. Hence the probability that the adversary wins in Game 1, but does not win in Game 0 is upper-bounded by $\varepsilon_{\mathsf{udlog}}(\lambda)$.

Game 2: This game is identical to Game 1 except it is additionally aborted if one of the opening is not accepting by an idealized verifier Verify'.

Game 1 to Game 2: The ideal verifier checks whether the following equality, for $\{\gamma_j\}_{j=1}^{l}, r$ picked at random, holds:

$$\sum_{j=0}^{l-1} r^j \left(\sum_{i \in K_j} \gamma_j^{i-1} \cdot \mathsf{f}_i(X) - \sum_{i \in K_j} \gamma_j^{i-1} \cdot s_{ij} \right) \quad \equiv \quad \sum_{j=0}^{l-1} r^j \mathsf{o}_\mathsf{j}(X)(X - z_j). \quad (4.1)$$

Since r has been picked as a random oracle output, probability that Eq. (4.1) holds while for some $j \in [0\,..\,l-1]$

$$r^j \left(\sum_{i \in K_j} \gamma^{i-1} \cdot \mathsf{f}_i(X) - \sum_{i \in K_j} \gamma^{i-1} \cdot s_{ij} \right) \not\equiv r^j \mathsf{o}_\mathsf{j}(X)(X - z_j)$$

is q/p cf. [91]. When $r^j \left(\sum_{i \in K_j} \gamma^{i-1} \cdot \mathsf{f}_i(X) - \sum_{i \in K_j} \gamma^{i-1} \cdot s_{ij} \right) = r^j \mathsf{o}_\mathsf{j}(X)(X - z_j)$ holds, polynomial $\mathsf{o}_\mathsf{j}(X)$ is uniquely determined from the uniqueness of polynomial composition.

Conclusion: We note that the idealized verifier $\mathsf{ve}(X)$ does not accept two different openings of a correct evaluation. Hence the probability that the adversary wins Game 2 is 0 and the probability that the adversary wins in Game 0 is upper-bounded by $\varepsilon_{\mathsf{udlog}}(\lambda) + \frac{q}{p}$. □

4.5 Concrete SNARKs Preliminaries

We refer the reader to Section 2.6 for general notation and the definition of bilinear groups. For pairing operation, we use additive notation in this work.

4.5.1 Algebraic Group Model

The algebraic group model (AGM) of Fuchsbauer, Kiltz, and Loss [89] lies somewhat between the standard and generic bilinear group model. In the AGM it is assumed that an adversary $\mathcal{A}$ can output a group element $[y] \in \mathbb{G}$ if $[y]$ has been computed by applying group operations to group elements given to $\mathcal{A}$ as input. It is further assumed, that $\mathcal{A}$ knows how to "build" $[y]$ from those elements. More precisely, the AGM requires that whenever $\mathcal{A}([\boldsymbol{x}])$ outputs a group element $[y]$ then it also outputs $\boldsymbol{c}$ such that $[y] = \boldsymbol{c}^\top \cdot [\boldsymbol{x}]$. Plonk, Sonic and Marlin have been shown secure using the AGM. An adversary that works in the AGM is called *algebraic*.

Ideal Verifier and Verification Equations. Let $(\mathsf{SRS}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ be a proof system. Observe that the SRS algorithms provide an SRS which can be interpreted as a set of group representation of polynomials evaluated at trapdoor elements. That is, for a trapdoor χ the SRS contains

$[\mathsf{p}_1(\chi), \ldots, \mathsf{p}_k(\chi)]_1$, for some polynomials $\mathsf{p}_1(X), \ldots, \mathsf{p}_k(X) \in \mathbb{F}_p[X]$. The verifier V accepts a proof π for instance x if (a set of) verification equation $\mathsf{ve}_{\mathsf{x},\pi}$ (which can also be interpreted as a polynomial in $\mathbb{F}_p[X]$ whose coefficients depend on messages sent by the prover) zeroes at χ. Following [91] we call verifiers who check that $\mathsf{ve}_{\mathsf{x},\pi}(\chi) = 0$ *real verifiers* as opposed to *ideal verifiers* who accept only when $\mathsf{ve}_{\mathsf{x},\pi}(X) = 0$. That is, while a real verifier accepts when a polynomial *evaluates* to zero, an ideal verifier accepts only when the polynomial *is* zero.

Although ideal verifiers are impractical, they are very useful in our proofs. More precisely, we show that the idealized verifier accepts an incorrect proof (what "incorrect" means depends on the situation) with at most negligible probability (and in many cases—never); when the real verifier accepts, but not the idealized one, then a malicious prover can be used to break the underlying security assumption (in our case—a variant of dlog.)

Analogously, idealized verifier can be defined for polynomial commitment schemes.

4.5.2 Dlog Assumptions in Standard and Updatable Setting

Definition 4.5.1 ((q_1, q_2)-dlog assumption)**.** *Let $\mathcal{A}$ be a* PPT *adversary that gets as input $[1, \chi, \ldots, \chi^{q_1}]_1, [1, \chi, \ldots, \chi^{q_2}]_2$, for some randomly picked $\chi \in \mathbb{F}_p$, the assumption requires that $\mathcal{A}$ cannot compute χ. That is*

$$\Pr[\chi = \mathcal{A}([1, \chi, \ldots, \chi^{q_1}]_1, [1, \chi, \ldots, \chi^{q_2}]_2) \mid \chi \leftarrow\$ \mathbb{F}_p] \leq \mathsf{negl}(\lambda).$$

Since all our protocols and security notions are in the updatable setting, it is natural to define the dlog assumptions also in the updatable setting. That is, instead of being given a dlog challenge the adversary $\mathcal{A}$ is given access to an update oracle as defined in Fig. 4.1. The honestly generated SRS is set to be a dlog challenge and the update algorithm Upd re-randomizing the challenge. We define this assumptions and show a reduction between the assumptions in the updatable and standard setting.

Note that for clarity we here refer to the SRS by Ch. Further, to avoid cluttering notation, we do not make the update proofs explicit. They are generated in the same manner as the proofs in Fig. 4.4.

Definition 4.5.2 ((q_1, q_2)-udlog assumption)**.** *Let $\mathcal{A}$ be a* PPT *adversary that gets oracle access to* UpdO *with internal algorithms* $(\mathsf{GenSRS}, \mathsf{Upd}, \mathsf{VerifySRS})$*, where* GenSRS *and* Upd *are defined as follows:*

- $\mathsf{GenSRS}(\lambda)$ *samples* $\chi \leftarrow\$ \mathbb{F}_p$ *and defines* $\mathsf{Ch} := ([1, \chi, \ldots, \chi^{q_1}]_1, [1, \chi, \ldots, \chi^{q_2}]_2)$.
- $\mathsf{Upd}(\mathsf{Ch}, \{\rho_j\}_{j=1}^n)$ *parses* Ch *as* $([\{A_i\}_{i=0}^{q_1}]_1, [\{B_i\}_{i=0}^{q_2}]_2)$*, samples* $\widetilde{\chi} \leftarrow\$ \mathbb{F}_p$*, and defines* $\widetilde{\mathsf{Ch}} := ([\{\widetilde{\chi}^i A_i\}_{i=0}^{q_1}]_1, [\{\widetilde{\chi}^i B_i\}_{i=0}^{q_2}]_2)$.

Then $\Pr\left[\bar{\chi} \leftarrow \mathcal{A}^{\mathsf{UpdO}}(\lambda)\right] \leq \mathsf{negl}(\lambda)$*, where* $([\{\bar{\chi}^i\}_{i=0}^{q_1}]_1, [\{\bar{\chi}^i\}_{i=0}^{q_2}]_2)$ *is the final* Ch.

Remark 4.5.3 (Single adversarial updates after an honest setup.). As an alternative to the updatable setting defined in Fig. 4.1, one can consider a slightly different model of setup, where the adversary is given an initial honestly-generated SRS and is then allowed to perform a malicious update in one-shot fashion. Groth et al. show in [120] that the two definitions are equivalent for polynomial commitment based SNARKs. We use this simpler definition in our reductions.

We show a reduction from (q_1, q_2)-dlog assumption to its variant in the updatable setting (with single adversarial update).

Lemma 4.5.4. (q_1, q_2)-dlog $\Rightarrow$ (q_1, q_2)-udlog.

Proof. We show a reduction $\mathcal{R}$ that uses an adversary $\mathcal{A}$ that breaks (q_1, q_2)-udlog to break (q_1, q_2)-dlog. Specifically, $\mathcal{R}$ proceeds as follows: given a dlog instance Ch as input, it sets Ch to be the initial (honestly generated) challenge and runs $\mathcal{A}$. After $\mathcal{A}$ performs its update and finalizes the dlog challenge it returns the answer χ'. Let $\chi_{\mathcal{A}}$ be the trapdoor contribution of adversary $\mathcal{A}$ in its update. Reduction $\mathcal{R}$ can extract this value from the update proof of $\mathcal{A}$. $\mathcal{R}$ now returns $\chi = \chi'\chi_{\mathcal{A}}^{-1}$ as the discrete logarithm of the original challenge Ch. □

Generalized Forking Lemma Although dubbed "general", the forking lemma of [19] is not general enough for our purpose as it is useful only for protocols where a witness can be extracted from just two transcripts. To be able to extract a witness from, say, an execution of $\mathbf{P}$ we need at least $(3\mathsf{n} + 6)$ valid proofs (where n is the number of constrains), $(\mathsf{n} + 1)$ for $\mathbf{S}$, and $2\mathsf{n} + 3$ for $\mathbf{M}$. Here we use a result by Attema et al. [12][6] which lower-bounds the probability of generating a tree of accepting transcripts T. We restate their Proposition 2 in our notation:

Lemma 4.5.5 (Run Time and Success Probability)**.** *Let $N = n_1 \cdot \dots \cdot n_\mu$ and $p = 2^{\Omega(\lambda)}$. Let $\varepsilon_{\mathsf{err}}(\lambda) = 1 - \prod_{i=1}^{\mu}\left(1 - \frac{n_i - 1}{p}\right)$. Assume adversary $\mathcal{A}$ that makes up to q random oracle queries and outputs an accepting proof with probability at least acc. There exists a tree building algorithm $\mathcal{T}$ for $(n_1, \dots, n_\mu)$-trees that succeeds in building a tree of accepting transcripts in expected running time $N + q(N - 1)$ with probability at least*

$$\frac{\mathsf{acc} - (q+1)\varepsilon_{\mathsf{err}}(\lambda)}{1 - \varepsilon_{\mathsf{err}}(\lambda)}.$$

Opening Uniqueness of Batched Polynomial Commitment Openings To show the unique response property required by our main theorem we show that the polynomial commitment schemes employed by concrete proof systems have unique openings, which, intuitively, assures that there is only one valid opening for a given committed polynomial and evaluation point:

Definition 4.5.6 (Unique opening property)**.** *Let $m \in \mathbb{N}$ be the number of committed polynomials, $l \in \mathbb{N}$ number of evaluation points, $\boldsymbol{c} \in \mathbb{G}^m$ be the commitments, $\boldsymbol{z} \in \mathbb{F}_p^l$ be the arguments the polynomials are evaluated at, K_j set of indices of polynomials which are evaluated at z_j, $\boldsymbol{s_i}$ vector of evaluations of $\mathsf{f_i}$, and $\boldsymbol{o_j}, \boldsymbol{o'_j} \in \mathbb{F}_p^{K_j}$ be the commitment openings. Then for every PPT adversary $\mathcal{A}$*

$$\Pr\left[\begin{array}{c}\mathsf{Verify}(\mathsf{srs}, \boldsymbol{c}, \boldsymbol{z}, \boldsymbol{s}, \boldsymbol{o}) = 1, \\ \mathsf{Verify}(\mathsf{srs}, \boldsymbol{c}, \boldsymbol{z}, \boldsymbol{s}, \boldsymbol{o'}) = 1, \\ \boldsymbol{o} \neq \boldsymbol{o'}\end{array} \middle| \; (\boldsymbol{c}, \boldsymbol{z}, \boldsymbol{s}, \boldsymbol{o}, \boldsymbol{o'}) \leftarrow \mathcal{A}^{\mathsf{UpdO}}(\mathsf{max}) \right] \leq \mathsf{negl}(\lambda)\,.$$

We show that the polynomial commitment schemes of Plonk, Sonic, and Marlin satisfy this requirement in Section 4.4.1.

[6] An earlier versions had its own forking lemma generalization. Attema et al. has a better bound.

Remark 4.5.7. Fig. 4.5, cf. Section 4.4.1, presents efficient variants of KZG [123] polynomial commitment schemes used in Plonk, Sonic and Marlin that support batched verification. Algorithms Com, Op, Verify take vectors as input and receive an additional arbitrary auxiliary string. This adversarially chosen string only provides additional context for the computation of challenges and allows reconstruction of proof transcripts $\tilde{\pi}[0..i]$ for batch challenge computations. We treat auxiliary input implicitly in the definition above.

4.6 Non-malleability of Plonk

In this section, we show that $\mathbf{P}_{\mathsf{FS}}$ is simulation-extractable. To this end, we first use the unique opening property to show that $\mathbf{P}_{\mathsf{FS}}$ has the 3-UR property, cf. Lemma 4.6.1. Next, we show that $\mathbf{P}_{\mathsf{FS}}$ is rewinding-based knowledge sound. That is, given a number of accepting transcripts whose first 3 messages match, we can either extract a witness for the proven statement or use one of the transcripts to break the udlog assumption. This result is shown in the AGM, cf. Lemma 4.6.2. We then show that $\mathbf{P}_{\mathsf{FS}}$ is 3-programmable trapdoor-less ZK in the AGM, cf. Lemma 4.6.3.

Given rewinding-based knowledge soundness, 3-UR and trapdoor-less zero-knowledge of $\mathbf{P}_{\mathsf{FS}}$, we invoke Theorem 4.3.1 and conclude that $\mathbf{P}_{\mathsf{FS}}$ is simulation-extractable.

4.6.1 Plonk Protocol Description

The constraint system. Assume C is a fan-in two arithmetic circuit, whose fan-out is unlimited and has n gates and m wires ($\mathsf{n} \leq \mathsf{m} \leq 2\mathsf{n}$). The constraint system of Plonk is defined as follows:

- Let $\boldsymbol{V} = (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$, where $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in [1 \,..\, \mathsf{m}]^{\mathsf{n}}$. Entries $\boldsymbol{a}_i, \boldsymbol{b}_i, \boldsymbol{c}_i$ represent indices of left, right and output wires of the circuit's i-th gate.
- Vectors $\boldsymbol{Q} = (\boldsymbol{q_L}, \boldsymbol{q_R}, \boldsymbol{q_O}, \boldsymbol{q_M}, \boldsymbol{q_C}) \in (\mathbb{F}^{\mathsf{n}})^5$ are called *selector vectors*: (a) If the i-th gate is a multiplication gate then $\boldsymbol{q_L}_i = \boldsymbol{q_R}_i = 0$, $\boldsymbol{q_M}_i = 1$, and $\boldsymbol{q_O}_i = -1$. (b) If the i-th gate is an addition gate then $\boldsymbol{q_L}_i = \boldsymbol{q_R}_i = 1$, $\boldsymbol{q_M}_i = 0$, and $\boldsymbol{q_O}_i = -1$. (c) $\boldsymbol{q_C}_i = 0$ for multiplication and addition gates.[7]

We say that vector $\boldsymbol{x} \in \mathbb{F}^{\mathsf{m}}$ satisfies constraint system if for all $i \in [1 \,..\, \mathsf{n}]$

$$\boldsymbol{q_L}_i \cdot \boldsymbol{x}_{\boldsymbol{a}_i} + \boldsymbol{q_R}_i \cdot \boldsymbol{x}_{\boldsymbol{b}_i} + \boldsymbol{q_O} \cdot \boldsymbol{x}_{\boldsymbol{c}_i} + \boldsymbol{q_M}_i \cdot (\boldsymbol{x}_{\boldsymbol{a}_i} \boldsymbol{x}_{\boldsymbol{b}_i}) + \boldsymbol{q_C}_i = 0.$$

Public inputs $(\mathsf{x}_j)_{j=1}^{\mathsf{n}}$ are enforced by adding the constrains

$$\boldsymbol{a}_i = j, \boldsymbol{q_L}_i = 1, \boldsymbol{q_M}_i = \boldsymbol{q_R}_i = \boldsymbol{q_O}_i = 0, \boldsymbol{q_C}_i = -\mathsf{x}_j \,,$$

for some $i \in [1 \,..\, \mathsf{n}]$.

Algorithms rolled out Plonk argument system is universal. That is, it allows to verify computation of any arithmetic circuit which has up to n gates using a single SRS. However, to make computation efficient, for each circuit there is allowed a preprocessing phase which extends the SRS with circuit-related polynomial evaluations.

For the sake of simplicity of the security reductions presented in this paper, we include in the SRS only these elements that cannot be computed without knowing the secret trapdoor χ. The rest of the preprocessed input can be computed using these SRS elements. We thus let them to be computed by the prover, verifier, and simulator separately.

[7] The $\boldsymbol{q_C}_i$ selector vector is meant to encode (input independent) constants.

Plonk *SRS generating algorithm* $\mathsf{GenSRS}(\mathcal{R})$*:* The SRS generating algorithm picks at random $\chi \leftarrow\!\$\ \mathbb{F}_p$, computes and outputs $\mathsf{srs} = \left(\left[\{\chi^i\}_{i=0}^{\mathsf{n}+5}\right]_1, [\chi]_2\right)$.

Preprocessing: Let $H = \{\omega^i\}_{i=1}^{\mathsf{n}}$ be a (multiplicative) n-element subgroup of a field $\mathbb{F}$ compound of n-th roots of unity in $\mathbb{F}$. Let $\mathsf{L}_i(X)$ be the i-th element of an n-elements Lagrange basis. During the preprocessing phase polynomials $\mathsf{S}_{\mathsf{id}\mathsf{j}}, \mathsf{S}_{\sigma\mathsf{j}}$, for $\mathsf{j} \in [1\,..\,3]$, are computed:

$$\begin{array}{ll} \mathsf{S}_{\mathsf{id}1}(X) = X, & \mathsf{S}_{\sigma 1}(X) = \sum_{i=1}^{\mathsf{n}} \sigma(i)\mathsf{L}_i(X), \\ \mathsf{S}_{\mathsf{id}2}(X) = k_1 \cdot X, & \mathsf{S}_{\sigma 2}(X) = \sum_{i=1}^{\mathsf{n}} \sigma(\mathsf{n}+i)\mathsf{L}_i(X), \\ \mathsf{S}_{\mathsf{id}3}(X) = k_2 \cdot X, & \mathsf{S}_{\sigma 3}(X) = \sum_{i=1}^{\mathsf{n}} \sigma(2\mathsf{n}+i)\mathsf{L}_i(X). \end{array}$$

Coefficients k_1, k_2 are such that $H, k_1 \cdot H, k_2 \cdot H$ are different cosets of $\mathbb{F}^*$, thus they define $3 \cdot \mathsf{n}$ different elements. Gabizon et al. [91] notes that it is enough to set k_1 to a quadratic residue and k_2 to a quadratic non-residue.

Furthermore, we define polynomials $\mathsf{q}_\mathsf{L}, \mathsf{q}_\mathsf{R}, \mathsf{q}_\mathsf{O}, \mathsf{q}_\mathsf{M}, \mathsf{q}_\mathsf{C}$ such that

$$\begin{array}{ll} \mathsf{q}_\mathsf{L}(X) = \sum_{i=1}^{\mathsf{n}} \boldsymbol{q_L}_i \mathsf{L}_i(X), & \mathsf{q}_\mathsf{O}(X) = \sum_{i=1}^{\mathsf{n}} \boldsymbol{q_O}_i \mathsf{L}_i(X), \\ \mathsf{q}_\mathsf{R}(X) = \sum_{i=1}^{\mathsf{n}} \boldsymbol{q_R}_i \mathsf{L}_i(X), & \mathsf{q}_\mathsf{C}(X) = \sum_{i=1}^{\mathsf{n}} \boldsymbol{q_C}_i \mathsf{L}_i(X). \\ \mathsf{q}_\mathsf{M}(X) = \sum_{i=1}^{\mathsf{n}} \boldsymbol{q_M}_i \mathsf{L}_i(X), & \end{array}$$

Proving statements in $\mathbf{P}_{\mathsf{FS}}$ We show how prover's algorithm $\mathsf{P}(\mathsf{srs}, \mathsf{x} = (\mathsf{w}'_i)_{i=1}^{\mathsf{n}}, \mathsf{w} = (\mathsf{w}_i)_{i=1}^{3\cdot\mathsf{n}})$ operates for the Fiat–Shamir transformed version of Plonk. Note that for notational convenience w also contains the public input wires $\mathsf{w}'_i = \mathsf{w}_i$, $i \in [1\,..\,\ell]$.

Message 1 Sample $b_1, \ldots, b_9 \leftarrow\!\$\ \mathbb{F}_p$; compute $\mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X)$ as

$$\begin{array}{l} \mathsf{a}(X) = (b_1 X + b_2)\mathsf{Z}_\mathsf{H}(X) + \sum_{i=1}^{\mathsf{n}} \mathsf{w}_i \mathsf{L}_i(X) \\ \mathsf{b}(X) = (b_3 X + b_4)\mathsf{Z}_\mathsf{H}(X) + \sum_{i=1}^{\mathsf{n}} \mathsf{w}_{\mathsf{n}+i} \mathsf{L}_i(X) \\ \mathsf{c}(X) = (b_5 X + b_6)\mathsf{Z}_\mathsf{H}(X) + \sum_{i=1}^{\mathsf{n}} \mathsf{w}_{2\cdot\mathsf{n}+i} \mathsf{L}_i(X) \end{array}$$

Output polynomial commitments $[\mathsf{a}(\chi), \mathsf{b}(\chi), \mathsf{c}(\chi)]_1$.

Message 2 Compute challenges $\beta, \gamma \in \mathbb{F}_p$ by querying random oracle on partial proof, that is, $\beta = \mathcal{H}(\tilde{\pi}[0..1], 0)$, $\gamma = \mathcal{H}(\tilde{\pi}[0..1], 1)$.

Compute permutation polynomial $\mathsf{z}(X)$

$$\begin{array}{l} \mathsf{z}(X) = (b_7 X^2 + b_8 X + b_9)\mathsf{Z}_\mathsf{H}(X) + \mathsf{L}_1(X) + \\ + \sum_{i=1}^{\mathsf{n}-1} \left(\mathsf{L}_{i+1}(X) \prod_{j=1}^{i} \frac{(\mathsf{w}_j + \beta\omega^{j-1} + \gamma)(\mathsf{w}_{\mathsf{n}+j} + \beta k_1 \omega^{j-1} + \gamma)(\mathsf{w}_{2\mathsf{n}+j} + \beta k_2 \omega^{j-1} + \gamma)}{(\mathsf{w}_j + \sigma(j)\beta + \gamma)(\mathsf{w}_{\mathsf{n}+j} + \sigma(\mathsf{n}+j)\beta + \gamma)(\mathsf{w}_{2\mathsf{n}+j} + \sigma(2\mathsf{n}+j)\beta + \gamma)} \right) \end{array}$$

Output polynomial commitment $[\mathsf{z}(\chi)]_1$

Message 3 Compute the challenge $\alpha = \mathcal{H}(\tilde{\pi}[0..2])$, compute the quotient polynomial

$$\begin{array}{l} \mathsf{t}(X) = \\ (\mathsf{a}(X)\mathsf{b}(X)\mathsf{q}_\mathsf{M}(X) + \mathsf{a}(X)\mathsf{q}_\mathsf{L}(X) + \mathsf{b}(X)\mathsf{q}_\mathsf{R}(X) + \mathsf{c}(X)\mathsf{q}_\mathsf{O}(X) + \mathsf{PI}(X) + \mathsf{q}_\mathsf{C}(X))/\mathsf{Z}_\mathsf{H}(X) + \\ + ((\mathsf{a}(X) + \beta X + \gamma)(\mathsf{b}(X) + \beta k_1 X + \gamma)(\mathsf{c}(X) + \beta k_2 X + \gamma)\mathsf{z}(X))\alpha/\mathsf{Z}_\mathsf{H}(X) \\ - (\mathsf{a}(X) + \beta\mathsf{S}_{\sigma 1}(X) + \gamma)(\mathsf{b}(X) + \beta\mathsf{S}_{\sigma 2}(X) + \gamma)(\mathsf{c}(X) + \beta\mathsf{S}_{\sigma 3}(X) + \gamma)\mathsf{z}(X\omega))\alpha/\mathsf{Z}_\mathsf{H}(X) \\ + (\mathsf{z}(X) - 1)\mathsf{L}_1(X)\alpha^2/\mathsf{Z}_\mathsf{H}(X) \end{array}$$

Split $\mathsf{t}(X)$ into degree less then n polynomials $\mathsf{t}_{\mathsf{lo}}(X), \mathsf{t}_{\mathsf{mid}}(X), \mathsf{t}_{\mathsf{hi}}(X)$, such that $\mathsf{t}(X) = \mathsf{t}_{\mathsf{lo}}(X) + X^{\mathsf{n}}\mathsf{t}_{\mathsf{mid}}(X) + X^{2\mathsf{n}}\mathsf{t}_{\mathsf{hi}}(X)$. Output $[\mathsf{t}_{\mathsf{lo}}(\chi), \mathsf{t}_{\mathsf{mid}}(\chi), \mathsf{t}_{\mathsf{hi}}(\chi)]_1$.

Message 4 Get the challenge $\varsigma \in \mathbb{F}_p$, $\varsigma = \mathcal{H}(\tilde{\pi}[0..3])$. Compute opening evaluations $\mathsf{a}(\varsigma), \mathsf{b}(\varsigma), \mathsf{c}(\varsigma), \mathsf{S}_{\sigma 1}(\varsigma), \mathsf{S}_{\sigma 2}(\varsigma), \mathsf{t}(\varsigma), \mathsf{z}(\varsigma\omega)$, Compute the linearization polynomial

$$\mathsf{r}(X) = \begin{array}{l} \mathsf{a}(\varsigma)\mathsf{b}(\varsigma)\mathsf{q}_{\mathsf{M}}(X) + \mathsf{a}(\varsigma)\mathsf{q}_{\mathsf{L}}(X) + \mathsf{b}(\varsigma)\mathsf{q}_{\mathsf{R}}(X) + \mathsf{c}(\varsigma)\mathsf{q}_{\mathsf{O}}(X) + \mathsf{q}_{\mathsf{C}}(X) \\ + \alpha \cdot ((\mathsf{a}(\varsigma) + \beta\varsigma + \gamma)(\mathsf{b}(\varsigma) + \beta k_1\varsigma + \gamma)(\mathsf{c}(\varsigma) + \beta k_2\varsigma + \gamma) \cdot \mathsf{z}(X)) \\ - \alpha \cdot ((\mathsf{a}(\varsigma) + \beta\mathsf{S}_{\sigma 1}(\varsigma) + \gamma)(\mathsf{b}(\varsigma) + \beta\mathsf{S}_{\sigma 2}(\varsigma) + \gamma)\beta\mathsf{z}(\varsigma\omega) \cdot \mathsf{S}_{\sigma 3}(X)) \\ + \alpha^2 \cdot \mathsf{L}_1(\varsigma) \cdot \mathsf{z}(X) \end{array}$$

Output $\mathsf{a}(\varsigma), \mathsf{b}(\varsigma), \mathsf{c}(\varsigma), \mathsf{S}_{\sigma 1}(\varsigma), \mathsf{S}_{\sigma 2}(\varsigma), \mathsf{t}(\varsigma), \mathsf{z}(\varsigma\omega), \mathsf{r}(\varsigma)$.

Message 5 Compute the opening challenge $v \in \mathbb{F}_p$, $v = \mathcal{H}(\tilde{\pi}[0..4])$. Compute the openings for the polynomial commitment scheme

$$\mathsf{W}_{\varsigma}(X) = \frac{1}{X - \varsigma}\begin{pmatrix} \mathsf{t}_{\mathsf{lo}}(X) + \varsigma^{\mathsf{n}}\mathsf{t}_{\mathsf{mid}}(X) + \varsigma^{2\mathsf{n}}\mathsf{t}_{\mathsf{hi}}(X) - \mathsf{t}(\varsigma) + v(\mathsf{r}(X) - \mathsf{r}(\varsigma)) + v^2(\mathsf{a}(X) - \mathsf{a}(\varsigma)) \\ + v^3(\mathsf{b}(X) - \mathsf{b}(\varsigma)) + v^4(\mathsf{c}(X) - \mathsf{c}(\varsigma)) + v^5(\mathsf{S}_{\sigma 1}(X) - \mathsf{S}_{\sigma 1}(\varsigma)) \\ + v^6(\mathsf{S}_{\sigma 2}(X) - \mathsf{S}_{\sigma 2}(\varsigma)) \end{pmatrix}$$

$$\mathsf{W}_{\varsigma\omega}(X) = (\mathsf{z}(X) - \mathsf{z}(\varsigma\omega))/(X - \varsigma\omega)$$

Output $[\mathsf{W}_{\varsigma}(\chi), \mathsf{W}_{\varsigma\omega}(\chi)]_1$.

Plonk verifier $\mathsf{V}(\mathsf{srs}, \mathsf{x}, \pi)$**:**
The Plonk verifier works as follows

1. Validate all obtained group elements.
2. Validate all obtained field elements.
3. Parse the instance as $\{\mathsf{w}_i\}_{i=1}^{\mathsf{n}} \leftarrow \mathsf{x}$.
4. Compute challenges $\beta, \gamma, \alpha, \varsigma, v, u$ from the transcript.
5. Compute zero polynomial evaluation $\mathsf{Z}_{\mathsf{H}}(\varsigma) = \varsigma^{\mathsf{n}} - 1$.
6. Compute Lagrange polynomial evaluation $\mathsf{L}_1(\varsigma) = \frac{\varsigma^{\mathsf{n}} - 1}{\mathsf{n}(\varsigma - 1)}$.
7. Compute public input polynomial evaluation $\mathsf{PI}(\varsigma) = \sum_{i \in [1\,..\,\mathsf{n}]} \mathsf{w}_i\mathsf{L}_i(\varsigma)$.
8. Compute quotient polynomials evaluations

$$\mathsf{t}(\varsigma) = 1/\mathsf{Z}_{\mathsf{H}}(\varsigma) \cdot$$
$$\Big(\mathsf{r}(\varsigma) + \mathsf{PI}(\varsigma) - (\mathsf{a}(\varsigma) + \beta\mathsf{S}_{\sigma 1}(\varsigma) + \gamma)(\mathsf{b}(\varsigma) + \beta\mathsf{S}_{\sigma 2}(\varsigma) + \gamma)(\mathsf{c}(\varsigma) + \gamma)\mathsf{z}(\varsigma\omega)\alpha - \mathsf{L}_1(\varsigma)\alpha^2\Big) .$$

9. Compute batched polynomial commitment $[D]_1 = v\,[r]_1 + u\,[z]_1$ that is

$$[D]_1 = v\begin{pmatrix} \mathsf{a}(\varsigma)\mathsf{b}(\varsigma) \cdot [\mathsf{q}_{\mathsf{M}}]_1 + \mathsf{a}(\varsigma)\,[\mathsf{q}_{\mathsf{L}}]_1 + \mathsf{b}\,[\mathsf{q}_{\mathsf{R}}]_1 + \mathsf{c}\,[\mathsf{q}_{\mathsf{O}}]_1 + \\ + ((\mathsf{a}(\varsigma) + \beta\varsigma + \gamma)(\mathsf{b}(\varsigma) + \beta k_1\varsigma + \gamma)(\mathsf{c} + \beta k_2\varsigma + \gamma)\alpha + \mathsf{L}_1(\varsigma)\alpha^2) + \\ - (\mathsf{a}(\varsigma) + \beta\mathsf{S}_{\sigma 1}(\varsigma) + \gamma)(\mathsf{b}(\varsigma) + \beta\mathsf{S}_{\sigma 2}(\varsigma) + \gamma)\alpha\beta\mathsf{z}(\varsigma\omega)\,[\mathsf{S}_{\sigma 3}(\chi)]_1) \end{pmatrix} +$$
$$+ u\,[\mathsf{z}(\chi)]_1 .$$

10. Computes full batched polynomial commitment $[F]_1$:

$$
\begin{aligned}
[F]_1 = & \left([\mathsf{t}_{\mathsf{lo}}(\chi)]_1 + \varsigma^{\mathsf{n}}\,[\mathsf{t}_{\mathsf{mid}}(\chi)]_1 + \varsigma^{2\mathsf{n}}\,[\mathsf{t}_{\mathsf{hi}}(\chi)]_1\right) + u\,[\mathsf{z}(\chi)]_1 + \\
& + v \begin{pmatrix} \mathsf{a}(\varsigma)\mathsf{b}(\varsigma)\cdot[\mathsf{q}_{\mathsf{M}}]_1 + \mathsf{a}(\varsigma)\,[\mathsf{q}_{\mathsf{L}}]_1 + \mathsf{b}(\varsigma)\,[\mathsf{q}_{\mathsf{R}}]_1 + \mathsf{c}(\varsigma)\,[\mathsf{q}_{\mathsf{O}}]_1 + \\ + ((\mathsf{a}(\varsigma)+\beta\varsigma+\gamma)(\mathsf{b}(\varsigma)+\beta k_1\varsigma+\gamma)(\mathsf{c}(\varsigma)+\beta k_2\varsigma+\gamma)\alpha + \mathsf{L}_1(\varsigma)\alpha^2) + \\ - (\mathsf{a}(\varsigma)+\beta\mathsf{S}_{\sigma 1}(\varsigma)+\gamma)(\mathsf{b}(\varsigma)+\beta\mathsf{S}_{\sigma 2}(\varsigma)+\gamma)\alpha\beta\mathsf{z}(\varsigma\omega)\,[\mathsf{S}_{\sigma 3}(\chi)]_1) \end{pmatrix} \\
& + v^2\,[\mathsf{a}(\chi)]_1 + v^3\,[\mathsf{b}(\chi)]_1 + v^4\,[\mathsf{c}(\chi)]_1 + v^5\,[\mathsf{S}_{\sigma 1}(\chi)]_1 + v^6\,[\mathsf{S}_{\sigma 2}(\chi)]_1\ .
\end{aligned}
$$

11. Compute group-encoded batch evaluation $[E]_1$

$$
\begin{aligned}
[E]_1 = & \frac{1}{\mathsf{Z}_{\mathsf{H}}(\varsigma)} \begin{bmatrix} \mathsf{r}(\varsigma) + \mathsf{PI}(\varsigma) + \alpha^2\mathsf{L}_1(\varsigma) + \\ - \alpha\left((\mathsf{a}(\varsigma)+\beta\mathsf{S}_{\sigma 1}(\varsigma)+\gamma)(\mathsf{b}(\varsigma)+\beta\mathsf{S}_{\sigma 2}(\varsigma)+\gamma)(\mathsf{c}(\varsigma)+\gamma)\mathsf{z}(\varsigma\omega)\right) \end{bmatrix}_1 \\
& + \left[v\mathsf{r}(\varsigma) + v^2\mathsf{a}(\varsigma) + v^3\mathsf{b}(\varsigma) + v^4\mathsf{c}(\varsigma) + v^5\mathsf{S}_{\sigma 1}(\varsigma) + v^6\mathsf{S}_{\sigma 2}(\varsigma) + u\mathsf{z}(\varsigma\omega)\right]_1\ .
\end{aligned}
$$

12. Check whether the verification equation holds

$$
\begin{aligned}
&([\mathsf{W}_{\varsigma}(\chi)]_1 + u\cdot[\mathsf{W}_{\varsigma\omega}(\chi)]_1)\bullet[\chi]_2 - \\
&\qquad (\varsigma\cdot[\mathsf{W}_{\varsigma}(\chi)]_1 + u\varsigma\omega\cdot[\mathsf{W}_{\varsigma\omega}(\chi)]_1 + [F]_1 - [E]_1)\bullet[1]_2 = 0\,. \quad (4.2)
\end{aligned}
$$

The verification equation is a batched version of the verification equation from [123] which allows the verifier to check openings of multiple polynomials in two points (instead of checking an opening of a single polynomial at one point).

Plonk simulator $\mathsf{Sim}_{\chi}(\mathsf{srs}, \mathsf{td} = \chi, \mathsf{x})$**:** We describe the simulator in Lemma 4.6.3.

4.6.2 Simulation extractability of Plonk

Unique Response Property

Lemma 4.6.1. *Let* $\mathbf{PC}_{\mathbf{P}}$ *be a polynomial commitment that is* $\varepsilon_{\mathsf{bind}}(\lambda)$*-binding and has unique opening property with loss* $\varepsilon_{\mathsf{op}}(\lambda)$*. Then* $\mathbf{P}_{\mathsf{FS}}$ *is* 3-UR *against algebraic adversaries, who makes up to* q *random oracle queries, with security loss* $\varepsilon_{\mathsf{bind}}(\lambda) + \varepsilon_{\mathsf{op}}(\lambda)$*.*

Intuition. We show that an adversary who can break the 3-unique response property of $\mathbf{P}_{\mathsf{FS}}$ can be either used to break the commitment scheme's evaluation binding or unique opening property. The former happens with the probability upper-bounded by $\varepsilon_{\mathsf{bind}}(\lambda)$, the latter with the probability upper bounded by $\varepsilon_{\mathsf{op}}(\lambda)$.

Proof. Let $\mathcal{A}$ be an algebraic adversary tasked to break the 3-UR-ness of $\mathbf{P}_{\mathsf{FS}}$. We show that the first three prover's messages determine, along with the verifiers challenges, the rest of it. We denote by π^0 and π^1 the two proofs that the adversary outputs. To distinguish polynomials and commitments which an honest prover would send in the proof from the polynomials and commitments computed by the adversary we write the latter using indices 0 and 1 (two indices as we have two transcripts), e.g. to describe the quotient polynomial provided by the adversary we write t^0 and t^1 instead of t as in the description of the protocol.

We note that since the unique response property requires from π^0 and π^1 that the first place they possibly differ is the 4-th prover's message, then the challenge ς, that is picked by the adversary after the 3-rd message is the same in both transcripts. This challenge determines the evaluation point of polynomials $\mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X), \mathsf{t}(X), \mathsf{z}(X)$ which commitments are already sent.

In its fourth message, the prover provides evaluations of the aforementioned polynomials, along with evaluations of publicly known polynomials $\mathsf{S}_{\sigma 1}(\varsigma), \mathsf{S}_{\sigma 2}(\varsigma)$, and evaluation of a linearization polynomial $\mathsf{r}(\varsigma)$.

Note that the adversary can output two accepting proofs that differ on their fourth message only if it either manages to break evaluation binding of one of the opening, or provides an incorrect opening which is accepted due to a batching error. Since the commitment scheme is evaluation binding with security loss $\varepsilon_{\mathsf{bind}}(\lambda)$, the adversary can make π^0 and π^1 differ on the fourth message with the same probability.

Next, assume that the transcripts are the same up to the fourth message, but differ at the fifth. In that message, the adversary provides openings of the evaluations. Since the unique opening property, the adversary can open the valid evaluation of a polynomial to two different values with probability at most $\varepsilon_{\mathsf{op}}(\lambda)$. (We note that for the KZG polynomial commitment scheme, as used in [91], $\varepsilon_{\mathsf{op}}(\lambda) \leq \varepsilon_{\mathsf{udlog}}(\lambda) + q/p$, cf. Lemma 4.4.1.)

By the union bound, the adversary is able to break the unique response property with probability upper bounded by $\varepsilon_{\mathsf{bind}}(\lambda) + \varepsilon_{\mathsf{op}}(\lambda)$. □

Rewinding-Based Knowledge Soundness

Lemma 4.6.2. $\mathbf{P}_{\mathsf{FS}}$ *is* $(3, 3\mathsf{n}+6)$*-rewinding-based knowledge sound against algebraic adversaries who make up to* q *random oracle queries with security loss*

$$\varepsilon_{\mathsf{ks}}(\lambda, \mathsf{acc}, q) \leq \left(1 - \frac{\mathsf{acc} - (q+1)\left(\frac{3\mathsf{n}+5}{p}\right)}{1 - \frac{3\mathsf{n}+5}{p}}\right) + (3\mathsf{n}+6) \cdot \varepsilon_{\mathsf{udlog}}(\lambda)\,,$$

Here acc *is a probability that the adversary outputs an accepting proof, and* $\varepsilon_{\mathsf{udlog}}(\lambda)$ *is security of* $(\mathsf{n}+5, 1)$-udlog *assumption.*

Intuition. We use Attema et al. [12, Proposition 2] to bound the probability that an algorithm $\mathcal{T}$ does not obtain a tree of accepting transcripts in an expected number of runs. This happens with probability at most

$$1 - \frac{\mathsf{acc} - (q+1)\left(\frac{3\mathsf{n}+5}{p}\right)}{1 - \frac{3\mathsf{n}+5}{p}}$$

Then we analyze the case that one of the proofs in the tree T outputted by $\mathcal{T}$ is not accepting by the ideal verifier. This discrepancy can be used to break an instance of an updatable dlog assumption which happens with probability at most $(3\mathsf{n}+6) \cdot \varepsilon_{\mathsf{udlog}}(\lambda)$.

Proof. Let $\mathcal{A}^{\mathcal{H},\mathsf{UpdO}}(1^\lambda; r)$ be the adversary who outputs (x, π) such that $\mathbf{P}_{\mathsf{FS}}.\mathsf{V}$ accepts the proof. Let $\mathcal{T}$ be a tree-building algorithm of Lemma 4.5.5 that outputs a tree T, and let $\mathsf{Ext}_{\mathsf{ss}}$ be an extractor that given the tree output by $\mathcal{T}$ reveals the witness for x. The main idea of the proof is to show that an adversary who breaks rewinding-based knowledge soundness can be used to break

a udlog-problem instance. The proof goes by game hops. Note that since the tree branches after $\mathcal{A}$'s 3-rd message, the instance x, commitments $[\mathsf{a}(\chi), \mathsf{b}(\chi), \mathsf{c}(\chi), \mathsf{z}(\chi), \mathsf{t}_{\mathsf{lo}}(\chi), \mathsf{t}_{\mathsf{mid}}(\chi), \mathsf{t}_{\mathsf{hi}}(\chi)]_1$, and challenges α, β, γ are the same in all the transcripts. Also, the tree branches after the third adversary's message where the challenge ς is presented, thus tree T is built using different values of ς. We consider the following games.

Game 0: In this game the adversary wins if it outputs a valid instance–proof pair (x, π), and the extractor $\mathsf{Ext}_{\mathsf{ss}}$ does not manage to output a witness w such that $\mathcal{R}(\mathsf{x}, \mathsf{w})$ holds.

Game 1: In this game the environment aborts the game if the tree building algorithm $\mathcal{T}$ fails in building a tree of accepting transcripts T.

Game 0 to Game 1: By Lemma 4.5.5 probability that Game 1 is aborted, while Game 0 is not, is at most

$$1 - \frac{\mathsf{acc} - (q+1)\left(\frac{3\mathsf{n}+5}{p}\right)}{1 - \frac{3\mathsf{n}+5}{p}}.$$

Game 2: In this game the environment additionally aborts if at least one of its proofs in T is not accepting by an ideal verifier.

Game 1 to Game 2: As usual, we show a reduction that breaks an instance of a udlog assumption when Game 2 is aborted, while Game 1 is not.

Let $\mathcal{R}_{\mathsf{udlog}}$ be a reduction that gets as input an $(\mathsf{n}+5, 1)$-udlog instance $[1, \ldots, \chi^{\mathsf{n}+5}]_1, [\chi]_2$. Then it can update the instance to another one $\left[1, \ldots, \chi'^{\mathsf{n}+5}\right]_1, [\chi']_2$. Eventually, the reduction outputs χ'. The reduction $\mathcal{R}_{\mathsf{udlog}}$ proceeds as follows. First, it builds $\mathcal{A}$'s SRS srs using the input udlog instance. Then it processes the adversary's update query by adding it to the list Q_{srs} and passing it to its own update oracle getting instance $\left[1, \ldots, \chi'^{\mathsf{n}+5}\right]_1, [\chi']_2$. The updated SRS srs' is then computed and given to $\mathcal{A}$. $\mathcal{R}_{\mathsf{udlog}}$ also takes care of the random oracle queries made by $\mathcal{A}$. It picks their answers honestly and write them in $Q_{\mathcal{H}}$. The reduction then starts $\mathcal{T}(\mathsf{srs}, \mathcal{A}, r, Q_{\mathcal{H}}, Q_{\mathsf{srs}})$.

Let $(1, \mathsf{T})$ be the output returned by $\mathcal{T}$. Let x be a relation proven in T. Consider a transcript $\pi \in \mathsf{T}$ such that $\mathsf{ve}_{\mathsf{x},\pi}(X) \neq 0$, but $\mathsf{ve}_{\mathsf{x},\pi}(\chi') = 0$. Since $\mathcal{A}$ is algebraic, all group elements included in T are extended by their representation as a combination of the input $\mathbb{G}_1$-elements. Hence, all coefficients of the verification equation polynomial $\mathsf{ve}_{\mathsf{x},\pi}(X)$ are known. Eventually, the reduction finds $\mathsf{ve}_{\mathsf{x},\pi}(X)$ zero points and returns χ' which is one of them.

Hence, the probability that the adversary wins in Game 2 but does not win in Game 1 is upper-bounded by $(3\mathsf{n}+6) \cdot \varepsilon_{\mathsf{udlog}}(\lambda)$.

Conclusion:

Note that the adversary can win in Game 2 only if $\mathcal{T}$ manages to produce a tree of accepting transcripts T, such that each of the transcripts in T is accepting by an ideal verifier. Note that since $\mathcal{T}$ produces $(3\mathsf{n}+6)$ accepting transcripts for different challenges ς, it obtains the same number of different evaluations of polynomials $\mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X), \mathsf{z}(X), \mathsf{t}(X)$. Since all the transcripts are accepting by an idealised verifier, the equality between polynomial $\mathsf{t}(X)$ and combination of polynomials $\mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X), \mathsf{z}(X)$ defined in prover's 3-rd message description holds. Hence, $\mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X)$ encodes the valid witness for the proven statement. $\mathsf{Ext}_{\mathsf{ss}}$ can recreate polynomials' coefficients by interpolation and reveal the witness given $(3\mathsf{n}+6)$ evaluations.

Hence, the probability that the adversary wins in Game 0 is upper-bounded by

$$\varepsilon_{\mathsf{ks}}(\lambda, \mathsf{acc}, q) \leq \left(1 - \frac{\mathsf{acc} - (q+1)\left(\frac{3\mathsf{n}+5}{p}\right)}{1 - \frac{3\mathsf{n}+5}{p}}\right) + (3\mathsf{n}+6) \cdot \varepsilon_{\mathsf{udlog}}(\lambda)\,.$$

□

Trapdoor-Less Zero-Knowledge of Plonk

Lemma 4.6.3. P_{FS} *is 3-programmable trapdoor-less zero-knowledge.*

Intuition. The simulator, that does not know the SRS trapdoor can make a simulated proof by programming the random oracle. It proceeds as follows. It picks a random witness and behaves as an honest prover up to the point when a commitment to the polynomial $\mathsf{t}(X)$ is sent. Since the simulator picked a random witness and $\mathsf{t}(X)$ is a polynomial only (modulo some negligible function) when the witness is correct, it cannot compute commitment to $\mathsf{t}(X)$ as it is a rational function. However, the simulator can pick a random challenge ς and a polynomial $\tilde{\mathsf{t}}(X)$ such that $\mathsf{t}(\varsigma) = \tilde{\mathsf{t}}(\varsigma)$. Then the simulator continues behaving as an honest prover. We argue that such a simulated proof is indistinguishable from a real one.

Proof. As noted in Section 4.2.1, subvertible zero-knowledge implies updatable zero-knowledge. Hence, here we show that Plonk is TLZK even against adversaries who picks the SRS on its own.

The adversary $\mathcal{A}(1^\lambda)$ picks an SRS srs and instance–witness pair (x, w) and gets a proof π simulated by the simulator Sim which proceeds as follows.

For its 1-st message the simulator picks randomly both the randomizers $b_1, \ldots, b_6$ and sets $\mathsf{w}_i = 0$ for $i \in [1\,..\,3\mathsf{n}]$. Then Sim outputs $[\mathsf{a}(\chi), \mathsf{b}(\chi), \mathsf{c}(\chi)]_1$. For the first challenge, the simulator picks permutation argument challenges β, γ randomly.

For its 2-nd message, the simulator computes $\mathsf{z}(X)$ from the newly picked randomizers b_7, b_8, b_9 and coefficients of polynomials $\mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X)$. Then it evaluates $\mathsf{z}(X)$ honestly and outputs $[\mathsf{z}(\chi)]_1$. Challenge α that should be sent by the verifier after the simulator's 2 message is picked by the simulator at random.

In its 3-rd message the simulator starts by picking at random a challenge ς, which in the real proof comes as a challenge from the verifier sent *after* the prover sends its 3-rd message. Then Sim computes evaluations $\mathsf{a}(\varsigma), \mathsf{b}(\varsigma), \mathsf{c}(\varsigma), \mathsf{S}_{\sigma 1}(\varsigma), \mathsf{S}_{\sigma 2}(\varsigma), \mathsf{PI}(\varsigma), \mathsf{L}_1(\varsigma), \mathsf{Z}_{\mathsf{H}}(\varsigma), \mathsf{z}(\varsigma\omega)$ and computes $\mathsf{t}(X)$ honestly. Since for a random $\mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X), \mathsf{z}(X)$ the constraint system is (with overwhelming probability) not satisfied and the constraints-related polynomials are not divisible by $\mathsf{Z}_{\mathsf{H}}(X)$, hence $\mathsf{t}(X)$ is a rational function rather than a polynomial. Then, the simulator evaluates $\mathsf{t}(X)$ at ς and picks randomly a degree-$(3\mathsf{n}+15)$ polynomial $\tilde{\mathsf{t}}(X)$ such that $\mathsf{t}(\varsigma) = \tilde{\mathsf{t}}(\varsigma)$ and publishes a commitment $[\tilde{\mathsf{t}}_{\mathsf{lo}}(\chi), \tilde{\mathsf{t}}_{\mathsf{mid}}(\chi), \tilde{\mathsf{t}}_{\mathsf{hi}}(\chi)]_1$. After that the simulator outputs ς as a challenge.

For the next message, the simulator computes polynomial $\mathsf{r}(X)$ as an honest prover would, cf. Section 4.6.1 and evaluates $\mathsf{r}(X)$ at ς.

The rest of the evaluations are already computed, thus Sim simply outputs $\mathsf{a}(\varsigma), \mathsf{b}(\varsigma), \mathsf{c}(\varsigma), \mathsf{S}_{\sigma 1}(\varsigma), \mathsf{S}_{\sigma 2}(\varsigma), \mathsf{t}(\varsigma), \mathsf{z}(\varsigma\omega)$. After that it picks randomly the challenge v, and prepares the the last message as an honest prover would. Eventually, Sim and outputs the final challenge, u, by picking it at random as well.

We argue about zero-knowledge as usual. The property holds since the polynomials that has witness elements at their coefficients are randomized by at least two randomizers and are evaluated at at most two points; and the simulator computes all polynomials as an honest prover would. □

Simulation Extractability of $\mathbf{P}_{\mathsf{FS}}$

Since Lemmas 4.6.1, 4.6.2, and 4.6.3 hold, $\mathbf{P}$ is 3-UR, rewinding-based knowledge sound and trapdoor-less zero-knowledge. We now make use of Theorem 4.3.1 and show that $\mathbf{P}_{\mathsf{FS}}$ is simulation-extractable as defined in 4.2.3.

Corollary 4.6.4 (Simulation extractability of $\mathbf{P}_{\mathsf{FS}}$)**.** $\mathbf{P}_{\mathsf{FS}}$ *is* updatable simulation-extractable *against any* PPT *adversary* $\mathcal{A}$ *who makes up to* q *random oracle queries and returns an accepting proof with probability at least* acc *with extraction failure probability*

$$\varepsilon_{\mathsf{se}}(\lambda, \mathsf{acc}, q) \leq \left(1 - \frac{\mathsf{acc} - \varepsilon_{\mathsf{ur}}(\lambda) - (q+1)\varepsilon_{\mathsf{err}}(\lambda)}{1 - \varepsilon_{\mathsf{err}}(\lambda)}\right) + (3\mathsf{n} + 6) \cdot \varepsilon_{\mathsf{udlog}}(\lambda),$$

where $\varepsilon_{\mathsf{err}}(\lambda) = \frac{3\mathsf{n}+5}{p}$*,* $\varepsilon_{\mathsf{ur}}(\lambda) \leq \varepsilon_{\mathsf{bind}}(\lambda) + \varepsilon_{\mathsf{op}}(\lambda)$*,* p *is the size of the field, and* n *is the number of constrains in the circuit.*

4.7 Non-malleability of Sonic

4.7.1 Preliminaries

Definition 4.7.1 ((q_1, q_2)-ldlog assumption)**.** *Let* $\mathcal{A}$ *be a* PPT *adversary that gets as input* $[\chi^{-q_1}, \ldots, 1, \ldots, \chi^{q_1}]_1, [\chi^{-q_2}, \ldots, 1, \ldots, \chi^{q_2}]_2$*, for some randomly picked* $\chi \in \mathbb{F}_p$*, the assumption requires that* $\mathcal{A}$ *cannot compute* χ*. That is*

$$\Pr\left[\chi = \mathcal{A}(\left[\chi^{-q_1}, \ldots, 1, \ldots, \chi^{q_1}\right]_1, \left[\chi^{-q_2}, \ldots, 1, \ldots, \chi^{q_2}\right]_2) \;\middle|\; \chi \leftarrow\!\$\, \mathbb{F}_p\right] \leq \mathsf{negl}(\lambda).$$

Definition 4.7.2 ((q_1, q_2)-uldlog assumption)**.** *Let* $\mathcal{A}$ *be a* PPT *adversary that gets oracle access to* UpdO *with internal algorithms* $(\mathsf{GenSRS}_{\mathsf{ldlog}}, \mathsf{Upd}_{\mathsf{ldlog}}, \mathsf{VerifySRS})$*, where* $\mathsf{GenSRS}_{\mathsf{ldlog}}$ *and* $\mathsf{Upd}_{\mathsf{ldlog}}$ *are defined as follows:*

- $\mathsf{GenSRS}_{\mathsf{ldlog}}(\lambda)$ *samples* $\chi \leftarrow\!\$\, \mathbb{F}_p$ *and defines*

$$\mathsf{Ch} := (\left[\chi^{-q_1}, \ldots, 1, \chi, \ldots, \chi^{q_1}\right]_1, \left[\chi^{-q_2}, \ldots, 1, \chi, \ldots, \chi^{q_2}\right]_2).$$

- $\mathsf{Upd}_{\mathsf{ldlog}}(\mathsf{Ch}, \{\rho_j\}_{j=1}^n)$ *parses* Ch *as* $\left(\left[\{A_i\}_{i=-q_1}^{q_1}\right]_1, \left[\{B_i\}_{i=-q_2}^{q_2}\right]_2\right)$*, samples* $\widetilde{\chi} \leftarrow\!\$\, \mathbb{F}_p$*, and defines* $\widetilde{\mathsf{Ch}} := \left(\left[\{\widetilde{\chi}^i A_i\}_{i=-q_1}^{q_1}\right]_1, \left[\{\widetilde{\chi}^i B_i\}_{i=-q_2}^{q_2}\right]_2\right)$.

Then

$$\Pr\left[\bar{\chi} \leftarrow \mathcal{A}^{\mathsf{UpdO}}(\lambda)\right] \leq \mathsf{negl}(\lambda),$$

where $\left(\left[\{\bar{\chi}^i\}_{i=-q_1}^{q_1}\right]_1, \left[\{\bar{\chi}^i\}_{i=-q_2}^{q_2}\right]_2\right)$ *is the finalized challenge.*

$\mathsf{GenSRS}(1^\lambda, \mathsf{max})$

$\alpha, \chi \leftarrow\$ \mathbb{F}_p^2$

return $\left[\{\chi^i\}_{i=-\mathsf{n}}^{\mathsf{n}}, \{\alpha\chi^i\}_{i=-\mathsf{n}, i\neq 0}^{\mathsf{n}}\right]_1, \left[\{\chi^i, \alpha\chi^i\}_{i=-\mathsf{n}}^{\mathsf{n}}\right]_2, [\alpha]_T$

$\mathsf{Com}(\mathsf{srs}, \mathsf{max}, \mathsf{f}(X))$

$\mathsf{c}(X) \leftarrow \alpha \cdot X^{\mathsf{d}-\mathsf{max}}\mathsf{f}(X)$

return $[c]_1 = [\mathsf{c}(\chi)]_1$

$\mathsf{Op}(\mathsf{srs}, z, s, f(X))$

$\mathsf{o}(X) \leftarrow \dfrac{\mathsf{f}(X) - \mathsf{f}(z)}{X - z}$

return $[\mathsf{o}(\chi)]_1$

$\mathsf{Verify}(\mathsf{srs}, \mathsf{max}, [c]_1, z, s, [\mathsf{o}(\chi)]_1)$

if $[\mathsf{o}(\chi)]_1 \bullet [\alpha\chi]_2 + [s - z\mathsf{o}(\chi)]_1 \bullet [\alpha]_2 = [c]_1 \bullet \left[\chi^{-\mathsf{d}+\mathsf{max}}\right]_2$ **then return** 1

else return 0.

Figure 4.6: $\mathsf{PC_S}$ polynomial commitment scheme.

4.7.2 Sonic Protocol Rolled-out

In this section we present Sonic's constraint system and algorithms. Reader familiar with them may jump directly to the next section.

The Constraint System Fig. 4.6 presents a variant of KZG [123] polynomial commitment schemes used in Sonic. Sonic's system of constraints composes of three n-long vectors $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ which corresponds to left and right inputs to multiplication gates and their outputs. It hence holds $\boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{c}$.

There is also Q linear constraints of the form

$$\boldsymbol{a}\boldsymbol{u_q} + \boldsymbol{b}\boldsymbol{v_q} + \boldsymbol{c}\boldsymbol{w_q} = k_q,$$

where $\boldsymbol{u_q}, \boldsymbol{v_q}, \boldsymbol{w_q}$ are vectors for the q-th linear constraint with instance value $k_q \in \mathbb{F}_p$. Furthermore define polynomials

$$\begin{aligned} \mathsf{u_i}(Y) &= \sum_{q=1}^{\mathsf{Q}} Y^{q+\mathsf{n}} u_{q,i}\,, \qquad & \mathsf{w_i}(Y) &= -Y^i - Y^{-i} + \sum_{q=1}^{\mathsf{Q}} Y^{q+\mathsf{n}} w_{q,i}\,, \\ \mathsf{v_i}(Y) &= \sum_{q=1}^{\mathsf{Q}} Y^{q+\mathsf{n}} v_{q,i}\,, \qquad & \mathsf{k}(Y) &= \sum_{q=1}^{\mathsf{Q}} Y^{q+\mathsf{n}} k_q. \end{aligned} \tag{4.3}$$

Sonic constraint system requires that

$$\boldsymbol{a}^\top \cdot \mathbf{u}(Y) + \boldsymbol{b}^\top \cdot \mathbf{v}(Y) + \boldsymbol{c}^\top \cdot \mathbf{w}(Y) + \sum_{i=1}^{\mathsf{n}} a_i b_i (Y^i + Y^{-i}) - \mathsf{k}(Y) = 0. \tag{4.4}$$

In Sonic we will use commitments to the following polynomials.

$$\begin{aligned} \mathsf{r}(X, Y) &= \sum_{i=1}^{\mathsf{n}} \left(a_i X^i Y^i + b_i X^{-i} Y^{-i} + c_i X^{-i-\mathsf{n}} Y^{-i-\mathsf{n}}\right) \\ \mathsf{s}(X, Y) &= \sum_{i=1}^{\mathsf{n}} \left(u_i(Y) X^{-i} + v_i(Y) X^i + w_i(Y) X^{i+\mathsf{n}}\right) \\ \mathsf{t}(X, Y) &= \mathsf{r}(X, 1)(\mathsf{r}(X, Y) + \mathsf{s}(X, Y)) - \mathsf{k}(Y)\,. \end{aligned}$$

Polynomials $\mathsf{r}(X,Y), \mathsf{s}(X,Y), \mathsf{t}(X,Y)$ are designed such that $\mathsf{t}(0,Y) = \boldsymbol{a}^\top \cdot \mathbf{u}(Y) + \boldsymbol{b}^\top \cdot \mathbf{v}(Y) + \boldsymbol{c}^\top \cdot \mathbf{w}(Y) + \sum_{i=1}^{\mathsf{n}} a_i b_i (Y^i + Y^{-i}) - \mathsf{k}(Y)$. That is, the prover is asked to show that $\mathsf{t}(0,Y) = 0$, cf. Eq. (4.4).

Furthermore, the commitment system in Sonic is designed such that it is infeasible for a PPT algorithm to commit to a polynomial with non-zero constant term.

Algorithms Rolled out Sonic *SRS generation* $\mathsf{GenSRS}(\mathcal{R})$. The SRS generating algorithm picks randomly $\alpha, \chi \leftarrow\!\!\$\ \mathbb{F}_p$ and outputs

$$\mathsf{srs} = \left(\left[\{\chi^i\}_{i=-\mathsf{d}}^{\mathsf{d}}, \{\alpha\chi^i\}_{i=-\mathsf{d}, i\neq 0}^{\mathsf{d}} \right]_1, \left[\{\chi^i, \alpha\chi^i\}_{i=-\mathsf{d}}^{\mathsf{d}} \right]_2, [\alpha]_T \right)$$

Sonic *prover* $\mathsf{P}(\mathsf{srs}, \mathsf{x}, \mathsf{w} = \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$.

Message 1 The prover picks randomly randomizers $c_{\mathsf{n}+1}, c_{\mathsf{n}+2}, c_{\mathsf{n}+3}, c_{\mathsf{n}+4} \leftarrow\!\!\$\ \mathbb{F}_p$. Sets $\mathsf{r}(X,Y) \leftarrow \mathsf{r}(X,Y) + \sum_{i=1}^{4} c_{\mathsf{n}+i} X^{-2\mathsf{n}-i}$. Commits to $\mathsf{r}(X,1)$ and outputs $[r]_1 \leftarrow \mathsf{Com}(\mathsf{srs}, \mathsf{n}, \mathsf{r}(X,1))$. Then it computes challenge $y = \mathcal{H}(\tilde{\pi}[0..1])$.

Message 2 P commits to $\mathsf{t}(X,y)$ and outputs $[t]_1 \leftarrow \mathsf{Com}(\mathsf{srs}, \mathsf{d}, \mathsf{t}(X,y))$. Then it gets a challenge $z = \mathcal{H}(\tilde{\pi}[0..2])$.

Message 3 The prover computes commitment openings. That is, it outputs

$$\begin{aligned} [o_a]_1 &= \mathsf{Op}(\mathsf{srs}, z, \mathsf{r}(z,1), \mathsf{r}(X,1)) \\ [o_b]_1 &= \mathsf{Op}(\mathsf{srs}, yz, \mathsf{r}(yz,1), \mathsf{r}(X,1)) \\ [o_t]_1 &= \mathsf{Op}(\mathsf{srs}, z, \mathsf{t}(z,y), \mathsf{t}(X,y)) \end{aligned}$$

along with evaluations $a' = \mathsf{r}(z,1), b' = \mathsf{r}(y,z), t' = \mathsf{t}(z,y)$. Then it engages in the signature of correct computation playing the role of the helper, i.e. it commits to $\mathsf{s}(X,y)$ and sends the commitment $[s]_1$, commitment opening

$$[o_s]_1 = \mathsf{Op}(\mathsf{srs}, z, \mathsf{s}(z,y), \mathsf{s}(X,y)),$$

and $s' = \mathsf{s}(z,y)$. Then it obtains a challenge $u = \mathcal{H}(\tilde{\pi}[0..3])$.

Message 4 For the next message the prover computes $[c]_1 \leftarrow \mathsf{Com}(\mathsf{srs}, \mathsf{d}, \mathsf{s}(u,Y))$ and computes commitments' openings

$$\begin{aligned} [w]_1 &= \mathsf{Op}(\mathsf{srs}, u, \mathsf{s}(u,y), \mathsf{s}(X,y)), \\ [q_y]_1 &= \mathsf{Op}(\mathsf{srs}, y, \mathsf{s}(u,y), \mathsf{s}(u,Y)), \end{aligned}$$

and returns $[w]_1, [q_y]_1, s = \mathsf{s}(u,y)$. Eventually the prover gets the last challenge $z' = \mathcal{H}(\tilde{\pi}[0..4])$.

Message 5 For the final message, P computes opening $[q_{z'}]_1 = \mathsf{Op}(\mathsf{srs}, z', \mathsf{s}(u,z'), \mathsf{s}(u,X))$ and outputs $[q_{z'}]_1$.

Sonic *verifier* $\mathsf{V}(\mathsf{srs}, \mathsf{x}, \pi)$. The verifier in Sonic runs as subroutines the verifier for the polynomial commitment. That is it sets $t' = a'(b' + s') - \mathsf{k}(y)$ and checks the following:

$$\begin{aligned}
&\mathbf{PC_S}.\mathsf{Verify}(\mathsf{srs}, \mathsf{n}, [r]_1, z, a', [o_a]_1), \\
&\mathbf{PC_S}.\mathsf{Verify}(\mathsf{srs}, \mathsf{n}, [r]_1, yz, b', [o_b]_1), \\
&\mathbf{PC_S}.\mathsf{Verify}(\mathsf{srs}, \mathsf{d}, [t]_1, z, t', [o_t]_1), \\
&\mathbf{PC_S}.\mathsf{Verify}(\mathsf{srs}, \mathsf{d}, [s]_1, z, s', [o_s]_1),
\end{aligned} \qquad \begin{aligned}
&\mathbf{PC_S}.\mathsf{Verify}(\mathsf{srs}, \mathsf{d}, [s]_1, u, s, [w]_1), \\
&\mathbf{PC_S}.\mathsf{Verify}(\mathsf{srs}, \mathsf{d}, [c]_1, y, s, [q_y]_1), \\
&\mathbf{PC_S}.\mathsf{Verify}(\mathsf{srs}, \mathsf{d}, [c]_1, z', \mathsf{s}(u, z'), [q_{z'}]_1),
\end{aligned}$$

and accepts the proof iff all the checks holds. Note that the value $\mathsf{s}(u, z')$ that is recomputed by the verifier uses separate challenges u and z'. This enables the batching of many proof and outsourcing of this part of the proof to an untrusted helper.

4.7.3 Unique Opening Property of $\mathbf{PC_S}$

Lemma 4.7.3. $\mathbf{PC_S}$ *has the unique opening property in the AGM.*

Proof. Let $z \in \mathbb{F}_p$ be the attribute the polynomial is evaluated at, $[c]_1 \in \mathbb{G}$ be the commitment, $s \in \mathbb{F}_p$ the evaluation value, and $o \in \mathbb{G}$ be the commitment opening. We need to show that for every PPT adversary $\mathcal{A}$ probability

$$\Pr\left[\begin{array}{l|l} \mathsf{Verify}(\mathsf{srs}, [c]_1, z, s, [o]_1) = 1, & \\ \mathsf{Verify}(\mathsf{srs}, [c]_1, z, \tilde{s}, [\tilde{o}]_1) = 1 & ([c]_1, z, s, [o]_1, [\tilde{o}]_1) \leftarrow \mathcal{A}^{\mathsf{UpdO}}(1^\lambda, \mathsf{max}) \end{array}\right]$$

is at most negligible.

As noted in [91, Lemma 2.2] it is enough to upper bound the probability of the adversary succeeding against the ideal verifier, who verifies equality between polynomials.

For a polynomial f, its degree upper bound max, evaluation point z, evaluation result s, and opening $[o(X)]_1$ the ideal verifier checks that

$$\alpha(X^{\mathsf{d}-\mathsf{max}} f(X) \cdot X^{-\mathsf{d}+\mathsf{max}} - s) \equiv \alpha \cdot o(X)(X - z)\,, \tag{4.5}$$

what is equivalent to

$$f(X) - s \equiv o(X)(X - z)\,. \tag{4.6}$$

Since $o(X)(X - z) \in \mathbb{F}_p[X]$ then from the uniqueness of polynomial composition, there is only one $o(X)$ that fulfills the equation above. □

4.7.4 Unique Response Property

The unique response property of $\mathbf{S_{FS}}$ follows from the unique opening property of the polynomial commitment scheme $\mathbf{PC_S}$.

Lemma 4.7.4. *If a polynomial commitment scheme* $\mathbf{PC_S}$ *is evaluation binding with security loss* $\varepsilon_{\mathsf{bind}}(\lambda)$ *and has unique openings property with security loss* $\varepsilon_{\mathsf{op}}(\lambda)$*, then* $\mathbf{S_{FS}}$ *is* 2-UR *against algebraic adversaries with security loss*

$$2 \cdot \varepsilon_{\mathsf{bind}}(\lambda) + \varepsilon_{\mathsf{op}}(\lambda).$$

Proof. Let $\mathcal{A}$ be an algebraic adversary tasked to break the 2-UR-ness of $\mathbf{S}_{\mathsf{FS}}$. We note that to show 2-UR-ness is is enough to show that the first prover's message determines, along with the verifiers challenges, the rest of it. We denote by π^0 and π^1 the two proofs for the same statement the adversary outputs. To distinguish polynomials and commitments which an honest prover sends in the proof from the polynomials and commitments computed by the adversary we write the latter using indices 0 and 1 (two indices as we have two transcripts), e.g. to describe the quotient polynomial provided by the adversary we write t^0 and t^1 instead of t as in the description of the protocol.

We note that since the unique response property requires from π^0 and π^1 that the first place they possibly differ is the 3-th prover's message, then the challenge z, that is picked by the adversary after the 2-rd message is the same in both transcripts. This challenge determines the evaluation point of polynomials $\mathsf{r}(X, 1), \mathsf{t}(X, y)$ which commitments are already sent.

In its third message, the prover provides evaluations of these polynomials along with their openings at z or yz. Note that the adversary can output two accepting proofs that differ on their third message only if it manages to break evaluation binding of one of the opening. Since the commitment scheme is evaluation binding with security loss $\varepsilon_{\mathsf{bind}}(\lambda)$, the adversary can make π^0 and π^1 differ on the third message with probability at most $\varepsilon_{\mathsf{bind}}(\lambda)$.

Similarly, in its fourth message, the prover provides an evaluation at u of polynomial $\mathsf{s}(X, y)$, an evaluation at y of $\mathsf{s}(u, Y)$, and the corresponding openings. Note that the adversary can output two accepting proofs that differ on their fourth message only if it manages to break evaluation binding of one of the opening. Since the commitment scheme is evaluation binding with security loss $\varepsilon_{\mathsf{bind}}(\lambda)$, the adversary can make π^0 and π^1 differ on the fourth message with probability at most $\varepsilon_{\mathsf{bind}}(\lambda)$.

Next, assume that the transcripts are the same up to the fourth message, but differ at the fifth. In that message, the adversary provides openings of the evaluations. Since the unique opening property, the adversary can open the valid evaluation of a polynomial to two different values with probability at most $\varepsilon_{\mathsf{op}}(\lambda)$.

By the union bound, the adversary is able to break the unique response property with probability upper bounded by $2\varepsilon_{\mathsf{bind}}(\lambda) + \varepsilon_{\mathsf{op}}(\lambda)$. □

4.7.5 Rewinding-Based Knowledge Soundness

Lemma 4.7.5. $\mathbf{S}_{\mathsf{FS}}$ *is* $(2, \mathsf{n}+1)$*-rewinding-based knowledge sound against algebraic adversaries who make up to* q *random oracle queries with security loss*

$$\varepsilon_{\mathsf{ks}}(\lambda, \mathsf{acc}, q) \leq \left(1 - \frac{\mathsf{acc} - (q+1)\left(\frac{\mathsf{n}}{p}\right)}{1 - \frac{\mathsf{n}}{p}}\right) + (\mathsf{n}+1) \cdot \varepsilon_{\mathsf{uldlog}}(\lambda)\,,$$

Here acc *is a probability that the adversary outputs an accepting proof, and* $\varepsilon_{\mathsf{uldlog}}(\lambda)$ *is the security of* $(\mathsf{max}, \mathsf{max})$-$\mathsf{uldlog}$ *assumption.*

Let $\mathcal{A}^{\mathcal{H},\mathsf{UpdO}}(1^\lambda; r)$ be the adversary who outputs (x, π) such that $\mathbf{S}_{\mathsf{FS}}.\mathsf{V}$ accepts the proof. Let $\mathcal{T}$ be the tree-building algorithm of Lemma 4.5.5 that outputs a tree T, and let $\mathsf{Ext}_{\mathsf{ss}}$ be an extractor that given the tree output by $\mathcal{T}$ reveals the witness for x. The main idea of the proof is to show that an adversary who breaks rewinding-based knowledge soundness can be used to break a uldlog-problem instance. The proof goes by game hops. Note that since the tree branches

after $\mathcal{A}$'s 2-nd message, the instance x, commitments $[\mathsf{r}(\chi,1),\mathsf{t}(\chi,y)]_1$, and challenge y are the same in all the transcripts. Also, the tree branches after the second adversary's message where the challenge z is presented, thus tree T is built using different values of z. We consider the following games.

Game 0: In this game, the adversary wins if it outputs a valid instance–proof pair (x,π), and the extractor $\mathsf{Ext}_{\mathsf{ss}}$ does not manage to output a witness w such that $\mathcal{R}(\mathsf{x},\mathsf{w})$ holds.

Game 1: In this game, the environment aborts the game if the tree building algorithm $\mathcal{T}$ fails in building a tree of accepting transcripts T.

Game 0 to Game 1: By Lemma 4.5.5 probability that Game 1 is aborted, while Game 0 is not, is, at most

$$1-\frac{\mathsf{acc}-(q+1)\left(\frac{\mathsf{n}}{p}\right)}{1-\frac{\mathsf{n}}{p}}\,.$$

Game 2: In this game the environment additionally aborts if at least one of its proofs in T is not accepting by an ideal verifier.

Game 1 to Game 2: As usual, we show a reduction that breaks an instance of a uldlog assumption when Game 2 is aborted, while Game 1 is not.

Let $\mathcal{R}_{\mathsf{uldlog}}$ be a reduction that gets as input an $(\mathsf{max},\mathsf{max})$-uldlog instance

$$\left[\chi^{-\mathsf{max}},\ldots,1,\ldots,\chi^{\mathsf{max}}\right]_1,\left[\chi^{-\mathsf{max}},\ldots,1,\ldots,\chi^{\mathsf{max}}\right]_2$$

. Then it can update the instance to another one $\left[\chi'^{-\mathsf{max}},\ldots,1,\ldots,\chi'^{\mathsf{max}}\right]_1,\left[\chi'^{-\mathsf{max}},\ldots,1,\ldots,\chi'^{\mathsf{max}}\right]_2$. Eventually, the reduction outputs χ'. The reduction $\mathcal{R}_{\mathsf{uldlog}}$ proceeds as follows. First, it builds $\mathcal{A}$'s SRS srs using the input uldlog instance. Then it processes the adversary's update query by adding it to the list Q_{srs} and passing it to its own update oracle getting instance $\left[\chi'^{-\mathsf{max}},\ldots,1,\ldots,\chi'^{\mathsf{max}}\right]_1,\left[\chi'^{-\mathsf{max}},\ldots,1,\ldots,\chi'^{\mathsf{max}}\right]_2$. The updated SRS srs' is then computed and given to $\mathcal{A}$. $\mathcal{R}_{\mathsf{uldlog}}$ also takes care of the random oracle queries made by $\mathcal{A}$. It picks their answers honestly and writes them in $Q_{\mathcal{H}}$. The reduction then starts $\mathcal{T}(\mathsf{srs},\mathcal{A},r,Q_{\mathcal{H}},Q_{\mathsf{srs}})$.

Let $(1,\mathsf{T})$ be the output returned by $\mathcal{T}$. Let x be a relation proven in T. Consider a transcript $\pi\in\mathsf{T}$ such that $\mathsf{ve}_{\mathsf{x},\pi}(X)\neq 0$, but $\mathsf{ve}_{\mathsf{x},\pi}(\chi')=0$. Since $\mathcal{A}$ is algebraic, all group elements included in T are extended by their representation as a combination of the input $\mathbb{G}_1$-elements. Hence, all coefficients of the verification equation polynomial $\mathsf{ve}_{\mathsf{x},\pi}(X)$ are known. Eventually, the reduction finds $\mathsf{ve}_{\mathsf{x},\pi}(X)$ zero points and returns χ' which is one of them.

Hence, the probability that the adversary wins in Game 2 but does not win in Game 1 is upper-bounded by $(\mathsf{n}+1)\cdot\varepsilon_{\mathsf{uldlog}}(\lambda)$.

Conclusion: Note that the adversary can win in Game 2 only if $\mathcal{T}$ manages to produce a tree of accepting transcript T, such that each of the transcripts in T is accepting by an ideal verifier. Note that since $\mathcal{T}$ produces $(\mathsf{n}+1)$ accepting transcripts for different challenges z, it obtains the same number of different evaluations of polynomial $\mathsf{t}(z,y)$ what allows to extract the witness, cf. [132].

Hence, the probability that the adversary wins in Game 0 is upper-bounded by

$$\varepsilon_{\mathsf{ks}}(\lambda,\mathsf{acc},q)\leq\left(1-\frac{\mathsf{acc}-(q+1)\left(\frac{\mathsf{n}}{p}\right)}{1-\frac{\mathsf{n}}{p}}\right)+(\mathsf{n}+1)\cdot\varepsilon_{\mathsf{uldlog}}(\lambda)\,.$$

4.7.6 Trapdoor-Less Zero-Knowledge of Sonic

Lemma 4.7.6. S_{FS} *is 2-programmable trapdoor-less zero-knowledge.*

Proof. The simulator proceeds as follows.

1. Pick randomly vectors $\boldsymbol{a}$, $\boldsymbol{b}$ and set
$$\boldsymbol{c} = \boldsymbol{a} \cdot \boldsymbol{b}. \tag{4.7}$$
2. Pick randomizers $c_{\mathsf{n}+1}, \ldots, c_{\mathsf{n}+4}$, honestly compute polynomials $\mathsf{r}(X,Y), \mathsf{r}'(X,Y), \mathsf{s}(X,Y)$ and pick randomly challenges y, z.
3. Output commitment $[r]_1 \leftarrow \mathsf{Com}(\mathsf{srs}, \mathsf{n}, \mathsf{r}(X,1))$ and challenge y.
4. Compute
$$\begin{aligned} a' &= \mathsf{r}(z,1), \\ b' &= \mathsf{r}(z,y), \\ s' &= \mathsf{s}(z,y). \end{aligned}$$
5. Pick polynomial $\mathsf{t}(X,Y)$ such that
$$\begin{aligned} &\mathsf{t}(X,y) = \mathsf{r}(X,1)(\mathsf{r}(X,y) + \mathsf{s}(X,y)) - \mathsf{k}(Y) \\ &\mathsf{t}(0,y) = 0 \end{aligned}$$
6. Output commitment $[t]_1 = \mathsf{Com}(\mathsf{srs}, \mathsf{d}, \mathsf{t}(X,y))$ and challenge z.
7. Continue following the protocol.

We note that the simulation is perfect. This comes since, except polynomial $\mathsf{t}(X,Y)$ all polynomials are computed following the protocol. For polynomial $\mathsf{t}(X,Y)$ we observe that in a case of both real and simulated proof the verifier only learns commitment $[t]_1 = \mathsf{t}(\chi, y)$ and evaluation $t' = \mathsf{t}(z,y)$. Since the simulator picks $\mathsf{t}(X,Y)$ such that

$$\mathsf{t}(X,y) = \mathsf{r}(X,1)(\mathsf{r}(X,y) + \mathsf{s}(X,y)) - \mathsf{k}(Y)$$

Values of $[t]_1$ are equal in both proofs. Furthermore, the simulator picks its polynomial such that $\mathsf{t}(0,y) = 0$, hence it does not need the trapdoor to commit to it. (Note that the proof system's SRS does not allow to commit to polynomials which have non-zero constant term). □

Remark 4.7.7. As noted in [132], Sonic is statistically subversion zero-knowledge (Sub-ZK). As noted in [4], one way to achieve subversion zero-knowledge is to utilize an extractor that extracts a SRS trapdoor from a SRS-generator. Unfortunately, a NIZK made subversion zero-knowledge by this approach cannot achieve perfect Sub-ZK as one has to count in the probability of extraction failure. However, with the simulation presented in Lemma 4.7.6, the trapdoor is not required for the simulator as it is able to simulate the execution of the protocol just by picking appropriate (honest) verifier's challenges. This result transfers to S_{FS}, where the simulator can program the random oracle to provide challenges that fits it.

4.7.7 Simulation Extractability of S_{FS}

Since Lemmas 4.7.4, 4.7.5, and 4.7.6 hold, S_{FS} is 2-UR, rewinding-based knowledge sound and trapdoor-less zero-knowledge. We now make use of Theorem 4.3.1 and show that S_{FS} is simulation-extractable as defined in 4.2.3.

Corollary 4.7.8 (Simulation extractability of S_{FS})**.** S_{FS} *is* updatable simulation-extractable *against any* PPT *adversary* $\mathcal{A}$ *who makes up to* q *random oracle queries and returns an accepting proof with probability at least* acc *with extraction failure probability*

$$\varepsilon_{\mathsf{se}}(\lambda, \mathsf{acc}, q) \leq \left(1 - \frac{\mathsf{acc} - \varepsilon_{\mathsf{ur}}(\lambda) - (q+1)\varepsilon_{\mathsf{err}}(\lambda)}{1 - \varepsilon_{\mathsf{err}}(\lambda)}\right) + (\mathsf{n}+1) \cdot \varepsilon_{\mathsf{uldlog}}(\lambda),$$

where $\varepsilon_{\mathsf{err}}(\lambda) = \frac{\mathsf{n}}{p}$*,* p *is the size of the field, and* n *is the number of constrains in the circuit.*

4.8 Non-malleability of Marlin

We show that Marlin is simulation-extractable. To that end, we show that Marlin has all the required properties: has unique response property, is rewinding-based knowledge sound, and its simulator can provide indistinguishable proofs without a trapdoor, just by programming the random oracle.

4.8.1 Marlin Protocol Rolled-out

Marlin uses R1CS as its arithmetization method. Given instance x, witness w and $|\mathsf{H}| \times |\mathsf{H}|$ matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$, the prover shows that $\boldsymbol{A}(\mathsf{x}^\top, \mathsf{w}^\top)^\top \circ \boldsymbol{B}(\mathsf{x}^\top, \mathsf{w}^\top)^\top = \boldsymbol{C}(\mathsf{x}^\top, \mathsf{w}^\top)^\top$, where $\circ$ denotes entry-wise product.

We assume that the matrices have at most $|\mathsf{K}|$ non-zero entries. Obviously, $|\mathsf{K}| \leq |\mathsf{H}|^2$. Let $b = 3$, the upper-bound of polynomial evaluations the prover has to provide for each of the sent polynomials. Denote by d an upper-bound for $\{|\mathsf{H}| + 2b - 1, 2|\mathsf{H}| + b - 1, 6|\mathsf{K}| - 6\}$.

The idea of showing that the constraint system is fulfilled is as follows. Denote by $\boldsymbol{z} = (\mathsf{x}, \mathsf{w})$. The prover computes polynomials $\mathsf{z}_{\mathsf{A}}(X), \mathsf{z}_{\mathsf{B}}(X), \mathsf{z}_{\mathsf{C}}(X)$ which encode vectors $\boldsymbol{Az}, \boldsymbol{Bz}, \boldsymbol{Cz}$ and have degree $< |\mathsf{H}|$. Importantly, when constraints are fulfilled, $\mathsf{z}_{\mathsf{A}}(X)\mathsf{z}_{\mathsf{B}}(X) - \mathsf{z}_{\mathsf{C}}(X) = \mathsf{h}_0(X)\mathsf{Z}_{\mathsf{H}}(X)$, for some $\mathsf{h}_0(X)$ and vanishing polynomial $\mathsf{Z}_{\mathsf{H}}(X)$. The prover sends commitments to these polynomials and shows that they have been computed correctly. More precisely, it shows that

$$\forall \boldsymbol{M} \in \{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}\}, \forall \kappa \in \mathsf{H}, \mathsf{z}_{\mathsf{M}}(\kappa) = \sum_{\iota \in \mathsf{H}} \boldsymbol{M}[\kappa, \iota]\mathsf{z}(\iota). \tag{4.8}$$

The ideal verifier checks the following equalities

$$\begin{aligned}
\mathsf{h}_3(\beta_3)\mathsf{Z}_{\mathsf{K}}(\beta_3) &= \mathsf{a}(\beta_3) - \mathsf{b}(\beta_3)(\beta_3\mathsf{g}_3(\beta_3) + \sigma_3/|\mathsf{K}|) \\
\mathsf{r}(\alpha, \beta_2)\sigma_3 &= \mathsf{h}_2(\beta_2)\mathsf{Z}_{\mathsf{H}}(\beta_2) + \beta_2\mathsf{g}2(\beta_2) + \sigma_2/|\mathsf{H}| \\
\mathsf{s}(\beta_1) + \mathsf{r}(\alpha, \beta_1)(\sum_M \eta_M \mathsf{z}_{\mathsf{M}}(\beta_1)) - \sigma_2\mathsf{z}(\beta_1) &= \mathsf{h}_1(\beta_1)\mathsf{Z}_{\mathsf{H}}(\beta_1) + \beta_1\mathsf{g}_1(\beta_1) + \sigma_1/|\mathsf{H}| \\
\mathsf{z}_{\mathsf{A}}(\beta_1)\mathsf{z}_{\mathsf{B}}(\beta_1) - \mathsf{z}_{\mathsf{C}}(\beta_1) &= \mathsf{h}_0(\beta_1)\mathsf{Z}_{\mathsf{H}}(\beta_1)
\end{aligned} \tag{4.9}$$

where $\mathsf{g}_{\mathsf{i}}(X), \mathsf{h}_{\mathsf{i}}(X)$, $i \in [1 \,..\, 3]$, $\mathsf{a}(X), \mathsf{b}(X), \sigma_1, \sigma_2, \sigma_3$ are polynomials and variables required by the sumcheck protocol which allows the verifier to efficiently verify that Eq. (4.8) holds.

4.8.2 Unique Response Property

Lemma 4.8.1. *Let* PC *be a commitment of knowledge that is evaluation binding with security loss* $\varepsilon_{\mathsf{bind}}(\lambda)$ *and has unique opening property with security loss* $\varepsilon_{\mathsf{op}}(\lambda)$*. Then* M_{FS} *is* 2-UR *against algebraic adversaries with security loss* $2 \cdot \varepsilon_{\mathsf{bind}}(\lambda) + \varepsilon_{\mathsf{op}}(\lambda)$*.*

Proof. The proof is similar to the proof of Lemma 4.6.1 and Lemma 4.7.4. An adversary who can break the 2-unique response property of M_{FS} can be either used to break the commitment scheme's evaluation binding or unique opening property. The former happens with the probability upper-bounded by $2 \cdot \varepsilon_{\mathsf{bind}}(\lambda)$, the latter with probability at most $\varepsilon_{\mathsf{op}}(\lambda)$. By the union bound, the adversary is able to break the unique response property with probability upper bounded by $2 \cdot \varepsilon_{\mathsf{bind}}(\lambda) + \varepsilon_{\mathsf{op}}(\lambda)$. □

4.8.3 Rewinding-Based Knowledge Soundness

Lemma 4.8.2. M_{FS} *is* $(2, 2\mathsf{n}{+}3)$*-rewinding-based knowledge sound against algebraic adversaries who make up to* q *random oracle queries with security loss*

$$\varepsilon_{\mathsf{ks}}(\lambda, \mathsf{acc}, q) \leq \left(1 - \frac{\mathsf{acc} - (q+1)\left(\frac{2\mathsf{n}+2}{p}\right)}{1 - \frac{2\mathsf{n}+2}{p}}\right) + (2\mathsf{n}+3) \cdot \varepsilon_{\mathsf{udlog}}(\lambda)\,,$$

Here acc *is a probability that the adversary outputs an acceptable proof, and* $\varepsilon_{\mathsf{udlog}}(\lambda)$ *is the security of* $(2\mathsf{n}+2, 1)$-udlog *assumption.*

Proof. The proof is similar to the proof of Lemma 4.6.2 and Lemma 4.7.5. We use Attema et al. [12, Proposition 2] to bound the probability that the tree-building algorithm $\mathcal{T}$ does not obtain a tree of acceptable transcript in an expected number of runs. This happens with probability at most

$$1 - \frac{\mathsf{acc} - (q+1)\left(\frac{2\mathsf{n}+2}{p}\right)}{1 - \frac{2\mathsf{n}+2}{p}}$$

Let T be the tree output by $\mathcal{T}$. If one of the proofs in T is not accepting by the ideal verifier, one can break an instance of an updatable dlog assumption which happens with probability at most $(2\mathsf{n}+3) \cdot \varepsilon_{\mathsf{udlog}}(\lambda)$. In the case that all the transcripts are accepting by the ideal verifier, but $\mathsf{Ext}_{\mathsf{ss}}$ fails to extract a valid witness from T, one can break the soundness of the ideal verifier in one of the transcripts. Taking a union bound completes the proof. □

4.8.4 Trapdoor-Less Zero-Knowledge of Marlin

Lemma 4.8.3. M_{FS} *is 2-programmable trapdoor-less zero-knowledge.*

Proof. The simulator follows the protocol except that it picks the challenges $\alpha, \eta_A, \eta_B, \eta_C, \beta_1, \beta_2, \beta_3$ before it picks the polynomials it sends.

First, it picks $\tilde{\mathsf{z}}_A(X)$, $\tilde{\mathsf{z}}_B(X)$ at random and $\tilde{\mathsf{z}}_C(X)$ such that $\tilde{\mathsf{z}}_A(\beta_1)\tilde{\mathsf{z}}_B(\beta_1) = \tilde{\mathsf{z}}_C(\beta_1)$. Given the challenges and polynomials $\tilde{\mathsf{z}}_A(X)$, $\tilde{\mathsf{z}}_B(X)$, $\tilde{\mathsf{z}}_C(X)$ the simulator computes $\sigma_1 \leftarrow \sum_{\kappa \in \mathsf{H}} \mathsf{s}(\kappa) + \mathsf{r}(\alpha, X)(\sum_{M \in \{A,B,C\}} \eta_M \tilde{\mathsf{z}}_M(X)) - \sum_{M \in \{A,B,C\}} \eta_M \mathsf{r}_M(\alpha, X)\tilde{\mathsf{z}}(X)$.

Then the simulator starts the protocol and follows it, except it programs the random oracle such that on partial transcripts, it returns the challenges already picked by Sim. □

4.8.5 Simulation Extractability of M_{FS}

Since Lemmas 4.8.1, 4.8.2, and 4.8.3 hold, M_{FS} is 2-UR, rewinding-based knowledge sound and trapdoor-less zero-knowledge. By making use of Theorem 4.3.1, we conclude that M_{FS} is simulation-extractable as defined in 4.2.3.

Corollary 4.8.4 (Simulation extractability of M_{FS})**.** M_{FS} *is* updatable simulation-extractable *against any* PPT *adversary* $\mathcal{A}$ *who makes up to* q *random oracle queries and returns an acceptable proof with probability at least* acc *with extraction failure probability*

$$\varepsilon_{\mathsf{se}}(\lambda, \mathsf{acc}, q) \leq \left(1 - \frac{\mathsf{acc} - \varepsilon_{\mathsf{ur}}(\lambda) - (q+1)\varepsilon_{\mathsf{err}}(\lambda)}{1 - \varepsilon_{\mathsf{err}}(\lambda)}\right) + (2\mathsf{n} + 3) \cdot \varepsilon_{\mathsf{udlog}}(\lambda),$$

where $\varepsilon_{\mathsf{err}}(\lambda) = \frac{2\mathsf{n}+2}{p}$*,* p *is the size of the field, and* n *is the number of constrains in the circuit.*

Chapter 5

Encryption to the Future

In this chapter we present our results on encryption to the future, first appeared in [49]. The contents of this chapter are taken almost verbatim from [49].

5.1 Introduction

Most cryptographic protocols assume that parties' identities are publicly known. This is a natural requirement, since standard secure channels are identified by a sender and a receiver. However, this status quo also makes it easy for adaptive (or proactive) adversaries to readily identify which parties are executing a protocol and decide on an optimal corruption strategy. In more practical terms, a party with a known identity (*e.g.* IP address) is at risk of being attacked.

A recent line of work [33, 102, 103] has investigated means for avoiding adaptive (or proactive) corruptions by having different randomly chosen committees of anonymous parties execute each round of a protocol. The rationale is that parties whose identities are unknown cannot be purposefully corrupted. Hence, having each round of a protocol executed by a fresh anonymous committee makes the protocol resilient to such powerful adversaries. However, this raises a new issue:

How can past committees efficiently transfer secret states to future yet-to-be-assigned anonymous committees?

5.1.1 Motivation: Role Assignment

The task of sending secret messages to a committee member that will be elected in the future can be abstracted as *role assignment*, a notion first introduced in [33] and further developed in [102]. This task consists of sending a message to an abstract role R at a given point in the future. A role is just a bit-string describing an abstract role, such as R ="party number j in round sl of the protocol Γ". Behind the scenes, there is a mechanism that samples the identity of a random party P_i and associates this machine to the role R. Such a mechanism allows anyone to send a message m to R and have m arrive at P_i chosen at some point in the future to act as R. A crucial point is: no one should know the identity of P_i even though P_i learns that it is chosen to act as R.

The approaches proposed in [33, 102, 103] for realizing role assignment all use an underlying Proof-of-Stake (PoS) blockchain (*e.g.* [73]). On a blockchain, a concrete way to implement role assignment is to sample a fresh key pair $(\mathsf{sk}_\mathsf{R}, \mathsf{pk}_\mathsf{R})$ for a public key encryption scheme, post

$(\mathsf{R}, \mathsf{pk}_\mathsf{R})$ on the blockchain and somehow send sk_R to a random P_i without leaking the identity of this party to anyone. Once $(\mathsf{R}, \mathsf{pk}_\mathsf{R})$ is known, every party has a target-anonymous channel to P_i and is able to encrypt under pk_R and post the ciphertext on the blockchain. Notice that using time-lock puzzles (or time-locked commitments/encryption) is not sufficient for achieving this notion, since only the party or parties elected for a role should receive a secret message encrypted for that role, while time-lock puzzles allow any party to recover the message if they invest enough computing time.

A shortcoming of the approaches of [33, 102, 103] is that, besides an underlying blockchain, they require an auxiliary committee to aid in generating $(\mathsf{sk}_\mathsf{R}, \mathsf{pk}_\mathsf{R})$ and selecting P_i. In the case of [33], the auxiliary committee performs cheap operations but can adversarially influence the probability distribution with which P_i is chosen. In the case of [102, 103], the auxiliary committee cannot bias this probability distribution but must perform very expensive operations (relying on Mix-Nets or FHE; see also Section 5.1.3). Moreover, these approaches have another caveat: they can only be used to select P_i to act as R according to a probability distribution *already known* at the time the auxiliary committee outputs $(\mathsf{R}, \mathsf{pk}_\mathsf{R})$. Hence, they only allow sending messages to future committees that have been *recently* elected. Later we explicitly consider this weaker setting—where we want to communicate with a "near-future" committee (i.e., whose distribution is known)—and dub it "Encryption to the Current Winner[1]" (ECW).

In this paper we further investigate solutions to the role-assignment problem[2]. Taking a step back from specific solutions for role assignment, we will focus on how to non-interactively *encrypt* to a future role with IND-CPA security *without* the aid of an auxiliary committee. We also discuss how to extend our approach to IND-CCA2 security and how to allow winners of a role to authenticate themselves when sending a message, both of which we can achieve with standard techniques.

5.1.2 Our Contributions

We look at the issue of sending messages to future roles as a problem on its own and introduce the Encryption to the Future (EtF) primitive as a central tool to solve it. Apart from defining this primitive and showing constructions based on previous works, we propose constructions based on new insights and investigate limits of EtF in different scenarios. Our general constructions for EtF work by lifting a weaker primitive, namely encryption for the aforementioned "near-future" setting, or ECW. Before providing further details, we summarize our contributions as follows:

[1]The word "winner" here refers to winning an assigned role and the underlying lottery of the PoS blockchain (see remainder of introduction).

[2]The family of protocols we consider actually has two role-related aspects to solve. The first—and the focus of this paper—is the aforementioned role assignment (RA) which deals with the sending of messages to future roles of a protocol while hiding the physical machine executing the role. The other aspect is *role execution* (RX) which focuses on the execution of the specific protocol that runs on top of the RA mechanism, i.e., what messages are sent to which roles and what specification the protocol implements. In [102] the so-called *You Only Speak Once* (YOSO) model is introduced for studying RX. In the YOSO model the protocol execution is between abstract roles which can each speak only once. Later these can then be mapped to physical machines using an RA mechanism. The work in [102] shows that once we can obtain RA in a synchronous model, then any well-formed ideal functionality can be implemented in the YOSO model with security against malicious, adaptive corruption of a minority of machines. Concretely, [102] gives an ideal functionality for RA and shows that a YOSO protocol for abstract roles can be compiled into the RA-hybrid model to give a protocol secure against adaptive attacks.

- A definition for the notion of *non-interactive* Encryption to the Future (EtF) in terms of an underlying blockchain and an associated lottery scheme that selects parties in the future to receive messages for a role. We study the strength of EtF as a primitive and prove that an EtF scheme—encryption towards parties selected at arbitrary points in the future—implies a flavor of witness encryption for NP.
- A novel construction of Encryption to the Current Winner (ECW), *i.e.* EtF where the receiver of a message is determined by the *current* state of the blockchain, which can be instantiated *without auxiliary committees* from standard assumptions via a construction based on generic primitives.
- A transformation from ECW to EtF through an auxiliary committee holding a *small* state, i.e., with communication complexity *independent* of plaintext size $|m|$ (in contrast to [33, 102, 103] where a committee's state grows with $|m|$).
- An application of ECW as a central primitive for realizing role assignment in protocols that require it (*e.g.* [33, 102, 103]).

Our EtF notion arguably provides a useful abstraction for the problem of transferring secret states to secret committees. Our ECW construction is the first primitive to realize role assignment without the need for an auxiliary committee. Moreover, building on new insights from our EtF notion and constructions, we show the first protocol for obtaining role assignment with no constraints on when parties are chosen to act as the role. While our protocol uses auxiliary committees, it improves on previous work requiring a communication complexity independent of the plaintext length.
We now elaborate on our results, discussing the intuition behind the notion of EtF, its constructions and its fundamental limits.

Encryption to the Future (EtF)—Section 5.3. As in previous works [33, 102, 103], an EtF scheme is defined with respect to an underlying PoS blockchain. We naturally use core features of the PoS setting to define what "future" means. The vast majority of PoS blockchains (*e.g.* [73]) associates a *slot number* to each block and are endowed with a lottery that selects parties to generate a block according to a stake distribution (*i.e.* the probability a party is selected is proportional to the stake the party controls). Thus, in EtF, we let a message be encrypted towards a party that is selected by the underlying blockchain's lottery scheme at a given future slot. We can generalize this and let the lottery select parties for multiple roles associated to each slot (so that committees consisting of multiple parties can be elected for a single point in time). An important point of our EtF definition is that it does not impose any constraints on the underlying blockchain's lottery scheme (*e.g.* it is not required to be anonymous) or on the slot when a party is supposed to be chosen to receive a message sent to a given role (*i.e.* party selection for a given role may happen w.r.t. a *future* stake distribution).

Relation to "Blockchain Witness Encryption" (BWE)—Section 5.8. We show that EtF implies a version of witness encryption [98] over a blockchain (similar to that of [112]). The crux of the proof: if we can encrypt a message towards a role assigned to a party only at an arbitrary point in the future then we can easily construct a witness encryption scheme exploiting EtF and a smart contract on the EtF's underlying blockchain. We also prove the opposite direction (BWE

implies EtF) showing that the notions are similar from a feasibility standpoint. But this also shows another important point: to implement non-interactive EtF, we would plausibly need strong assumptions (e.g., full-blown WE).

Encryption to the Current Winner (ECW)—Section 5.3. By the previous result we know that, unless we turn to strong assumptions, we may not construct a fully non-interactive EtF (i.e., without auxiliary committees); therefore, we look for efficient ways to construct EtF under standard assumptions while minimizing interaction. As a first step towards such a construction, we define the notion of Encryption to a Current Winner (ECW), which is a restricted version of EtF where messages can only be encrypted towards parties selected for a role whose lottery parameters are available for the *current* slot, the one in which we encrypt (this is as in previous constructions [33, 102, 103]).

Constructing ECW (non-interactively)—Section 5.5. We show that it is possible to construct a fully non-interactive ECW scheme from standard assumptions. Our construction relies on a milder flavor of witness encryption, which we call Witness Encryption over Commitments (cWE) and define it in Section 5.4. This primitive is significantly more restricted than full-fledged WE (see also discussion in Remark 5.4.1), but still powerful enough: we show in Section 5.5.1 that ECW can be constructed in a black-box manner from cWE, which in turn can be constructed from oblivious transfer and garbled circuits (Section 5.4.2). This construction improves over the previous results [33, 102, 103] since it does not rely on auxiliary committees.

Instantiating YOSO MPC using ECW—Section 5.6. The notion of ECW is more restricted than EtF, but it can still be useful in applications. We show how to use it as a building block for the YOSO MPC protocol of [102]. Here, each of the rounds in an MPC protocol is executed by a different committee. This same committee will simultaneously transfer its secret state to the next (near-future) committee, which in turn remains anonymous until it transfers its own secret state to the next committee, and so on. This setting clearly matches what ECW offers as a primitive, but it also introduces a few more requirements: 1. ECW ciphertexts must be non-malleable, *i.e.* we need an IND-CCA secure ECW scheme; 2. Only one party is selected for each role; 3. A party is selected for a role at random with probability proportional to its relative stake on the underlying PoS blockchain; 4. Parties selected for roles remain anonymous until they choose to reveal themselves; 5. A party selected for a role must be able to authenticate messages on behalf of the role, *i.e.* publicly proving that it was selected for a certain role and that it is the author of a message. We show that all of these properties can be obtained departing from an IND-CPA secure ECW scheme instantiated over a natural PoS blockchain (*e.g.* [73]). First, we observe that VRF-based lottery schemes implemented in many PoS blockchains are sufficient to achieve properties 1, 2 and 3. We then observe that natural block authentication mechanisms used in such PoS blockchains can be used to obtain property 4. Finally, we show that standard techniques can be used to obtain an IND-CCA secure ECW scheme from an IND-CPA secure ECW scheme.

Constructing EtF from ECW (interactively)—Section 5.7. Since we argued the implausibility of constructing EtF non-interactively from standard assumptions, we study how to transform an ECW scheme into an unrestricted EtF scheme when given access to an auxiliary committee but

Type	Scheme	Communication	Committee?	Interaction?
ECW	CaBKaS [33]	$O(1)$	yes	yes
	RPIR [103]	$O(1)$	yes	yes
	cWE (MS-NISC) (Sec. 5.4.2)	$O(N)$	no	no*
	cWE (GC+OT) (Sec. 5.4.2)	$O(N)$	no	no*
EtF	IBE (Sec. 5.7)	$O(1)$	yes	yes
	Full-fledged WE	$O(1)$	no	no

Figure 5.1: The column "Committee?" indicates whether a committee is required. The column "Communication" refers to whether the communication complexity grows or not with N, the number of all parties. We denote by an asterisk non-interactive solutions that require sending a first reusable message during the initial step.

with "low communication" (and still from standard assumptions). We explain what we mean by "low communication" by an example of its opposite: in previous works ([33, 102, 103]) successive committees were required to store and reshare secret shares of every message to be sent to a party selected in the future. That is, their communication complexity grows both with the number and the amount and length of the encrypted messages. In contrast, our solution has communication complexity independent of the plaintext length. How our transformation from ECW to EtF works: we associate each role in the future to a unique identity of an Identity Based Encryption scheme (IBE); to encrypt a message towards a role we apply the encryption of the IBE scheme. When, at any point in the future, a party for that role is selected, a committee generates and delivers the corresponding secret key for that role/identity. To realize the latter step, we apply YOSO MPC instantiated from ECW as shown in Section 5.6. In contrast to previous schemes, our auxiliary committee only needs to hold shares of the IBE's master secret key and so it performs communication/computation dependent on the security parameter but not on the length/amount of messages encrypted to the future.

5.1.3 Previous Works

We compare previous works related to our notions of EtF and ECW (encryption to future and current winner, respectively) in Fig. 5.1.

Encryption to the Current Winner (ECW). We recall that ECW is an easier setting than EtF: both the stake distribution and the randomness extracted from the blockchain are static and known at the time of encryption. This means that all of the parameters except the secret key of the lottery winner are available to the encryption algorithm. We now survey works that solved this problem and compare them to our solutions:

- "Can a Blockchain Keep a Secret?" (CaBKaS) [33]. The work of [33] addresses the setting where a dynamically changing committee (over a public blockchain) maintains a secret. The main challenge in order for the committee to *securely* reshare its secret can be summarized as: how to select a small committee from a large population of parties so that everyone can send secure messages to the committee members without knowing who they are? The solution of [33] is to select the "secret-holding" committee by having another committee, a "nominating committee", that nominates members of the former (while the members of the nominating committee are self-nominated).

One can see the nominating committee as a tool providing the ECW functionality. A major caveat in such a solution, however, is that to guarantee an honest majority in the committees, [33] can only tolerate up to 1/4 as the fraction of corrupted parties. This is because corrupted nominators can always select corrupted parties, whereas honest nominators may select corrupted parties by chance. We can improve this through our non-interactive ECW: we can remove the nominating committee and just let the current committee ECW-encrypt their secret shares to the roles of the next committee.

- "Random-Index PIR" (RPIR) [103]. The recent work of [103] defines a new flavour of Private Information Retrieval (PIR) called Random-index PIR (or RPIR) that allows each committee to perform the nomination task by themselves. While RPIR improves on [33] (not requiring a nominating committee and tolerating up to 1/2 of corrupted parties), its constructions are inefficient, either based on Mix-Nets or Fully Homomorphic Encryption (FHE). The construction based on Mix-Nets uses k shufflers, where k is the security parameter, and has an impractical communication complexity of $O(nk^2)$, where n is the number of public keys that each shuffler broadcasts. The FHE-based construction gives a total communication complexity of $O(k^3)$ where $O(k)$ is the length of an FHE decryption share.

WE over commitments (cWE). Benhamouda and Lin [29] defined a type of witness encryption, called "Witness Encryption for NIZK of Commitments". In their setting, parties first commit to their private inputs once and for all. Later, an encryptor can produce a ciphertext so that any party with a committed input that satisfies the relation (specified at encryption time) can decrypt. More accurately, who can decrypt is any party *with a NIZK showing that the committed input satisfies the relation*. The authors construct this primitive based on standard assumptions in asymmetric bilinear groups.

In our work, we generalize the encryption notion in [29], formalize it as cWE and finally use it to construct ECW. While the original construction of [29] fits the definition of cWE, we observe it is an overkill for our application. Specifically our setting does not require NIZKs to be involved in encryption/decryption. We instead give more efficient instantiations based on two-party Multi-Sender Non-Interactive Secure Computation (MS-NISC) protocols and Oblivious Transfer plus Garbled Circuits.

Encryption to the Future (EtF). The general notion of EtF is significantly harder to realize than ECW (as we show in Section 5.8). Below we discuss natural ideas to obtain EtF. They can be seen as illustrating two extremes where our approach (Section 5.7) lies in the middle.

- Non-Interactive—Using Witness Encryption [98]: One trivial approach to realize EtF is to use full-fledged general Witness Encryption [98] (WE) for the arithmetic relation $\mathcal{R}$ being the lottery predicate such that the party who holds a winning secret key sk can decrypt the ciphertext. However, constructing a general witness encryption scheme [98] which we can instantiate reliably is still an open problem. Existing constructions rely on very strong assumptions such as multilinear maps, indistinguishability obfuscation or other complexity theoretical conjectures [17]. The challenges in applying this straightforward solution are not surprising given our result showing that EtF implies a flavor of WE.

- Interactive—Multiple Committees and Continuous Executions of ECW: A simple way to achieve an interactive version of EtF is to first encrypt secret shares of a message towards

members of a committee that then re-share their secrets towards members of a future anonymous committee via an invocation of ECW (in our instantiations or those in [33] and [103]). This is essentially the solution proposed in CaBKaS [33] where committees interact in order to carry a secret (on the blockchain) into the future. Notice that, for a fixed security parameter and corruption ratio, the communication complexity of the protocol executed by the committee in this solution depends on the plaintext message length. On the other hand, for a fixed security parameter and corruption ratio, the communication complexity of our committee-based transformation from ECW to EtF is *constant*.

Other works. Using blockchains in order to construct non-interactive primitives with game-based security has been previously considered in [110]. Other approaches for transferring secret state to future committees have been proposed in [112], although anonymity is not a concern in this setting. On the other hand, using anonymity to overcome adaptive corruption has been proposed in [95], although this work considers anonymous channels among a fixed set of parties.

5.2 Preliminaries

5.2.1 Proof-of-Stake (PoS) Blockchains

In this work we rely on PoS-based blockchain protocols. In such a protocol, each participant is associated with some stake in the system. A process called leader election encapsulates a lottery mechanism that ensures (of all eligible parties) each party succeeds in generating the next block with probability proportional to its stake in the system. In order to formally argue about executions of such protocols, we depart from the framework presented in [110] which, in turn, builds on the analysis done in [96] and [142]. We invite the reader to re-visit the abstraction used in [110]. We present a summary of the framework in the full version [49] and discuss below the main properties we will use in the remainder of this paper. Moreover, we note that in [110] it is proven that there exist PoS blockchain protocols with the properties described below, *e.g.* Ouroboros Praos [73].

Blockchain Structure.

A genesis block $B_0 = \{(\mathsf{Sig.pk}_1, \mathsf{aux}_1, \mathsf{stake}_1), \ldots, (\mathsf{Sig.pk}_n, \mathsf{aux}_n, \mathsf{stake}_n), \mathsf{aux}\}$ associates each party P_i to a signature scheme public key $\mathsf{Sig.pk}_i$, an amount of stake stake_i and auxiliary information aux_i (*i.e.* any other relevant information required by the blockchain protocol, such as verifiable random function public keys). A blockchain $\mathbf{B}$ relative to a genesis block B_0 is a sequence of blocks $B_1, \ldots, B_n$ associated with a strictly increasing sequence of slots $\mathsf{sl}_1, \ldots, \mathsf{sl}_m$ such that $B_i = (\mathsf{sl}_j, H(B_{i-1}), \mathsf{d}, \mathsf{aux}))$. Here, sl_j indicates the time slot that B_i occupies, $H(B_{i-1})$ is a collision resistant hash of the previous block, d is data and aux is auxiliary information required by the blockchain protocol (*e.g.* a proof that the block is valid for slot sl_j). We denote by $\mathbf{B}^{\lceil \ell}$ the chain (sequence of blocks) $\mathbf{B}$ where the last ℓ blocks have been removed and if $\ell \geq |\mathbf{B}|$ then $\mathbf{B}^{\lceil \ell} = \epsilon$. Also, if $\mathbf{B}_1$ is a prefix of $\mathbf{B}_2$ we write $\mathbf{B}_1 \preceq \mathbf{B}_2$. Each party participating in the protocol has public identity P_i and most messages will be a transaction of the following form: $m = (P_i, P_j, \mathsf{q}, \mathsf{aux})$ where P_i transfers q coins to P_j along with some optional, auxiliary information aux.

Blockchain Setup and Key Knowledge.

As in [73], we assume that the genesis block is generated by an initialization functionality $\mathcal{F}_{\mathsf{INIT}}$ that registers all parties' keys. Moreover, we assume that primitives specified in separate functionalities in [73] as incorporated into $\mathcal{F}_{\mathsf{INIT}}$. $\mathcal{F}_{\mathsf{INIT}}$ is executed by the environment $\mathcal{Z}$ as defined below and is parameterized by a stake distribution associating each party P_i to an initial stake stake_i. Upon being activated by P_i for the first time, $\mathcal{F}_{\mathsf{INIT}}$ generates a signature key pair $\mathsf{Sig.sk}_i, \mathsf{Sig.pk}_i$, auxiliary information aux_i and a lottery witness $\mathsf{sk}_{L,i}$, which will be defined as part of the lottery predicate in Section 5.2.1, sending $(\mathsf{Sig.sk}_i, \mathsf{Sig.pk}_i, \mathsf{aux}_i, \mathsf{sk}_{L,i}, \mathsf{stake}_i)$ to P_i as response. After all parties have activated $\mathcal{F}_{\mathsf{INIT}}$, it responds to requests for a genesis block by providing $B_0 = \{(\mathsf{Sig.pk}_1, \mathsf{aux}_1, \mathsf{stake}_1), \ldots, (\mathsf{Sig.pk}_n, \mathsf{aux}_n, \mathsf{stake}_n), \mathsf{aux}\}$, where aux is generated according to the underlying blockchain consensus protocol.

Since $\mathcal{F}_{\mathsf{INIT}}$ generates keys for all parties, we capture the fact that even corrupted parties have registered public keys and auxiliary information such that they know the corresponding secret keys. Moreover, when our EtF constructions are used as part of more complex protocols, a simulator executing the EtF and its underlying blockchain with the adversary will be able to predict which ciphertexts can be decrypted by the adversary by simulating $\mathcal{F}_{\mathsf{INIT}}$ and learning these keys. This fact will be important when arguing the security of protocols that use our notion of EtF.

Evolving Blockchains.

In order to define an EtF scheme, some concept of future needs to be established. In particular we want to make sure that the initial chain $\mathbf{B}$ has "correctly" evolved into the final chain $\tilde{\mathbf{B}}$. Otherwise, the adversary can easily simulate a blockchain where it wins a future lottery and finds itself with the ability to decrypt. Fortunately, the *Distinguishable Forking* property provides just that (see [110] for more details). A sufficiently long chain in an honest execution can be distinguished from a fork generated by the adversary by looking at the combined amount of stake proven in such a sequence of blocks. We encapsulate this property in a predicate called $\mathsf{evolved}(\cdot,\cdot)$. First, let $\Gamma^V = (\mathsf{UpdateState}^V, \mathsf{GetRecords}, \mathsf{Broadcast})$ be a blockchain protocol with validity predicate V and where the $(\alpha, \beta, \ell_1, \ell_2)$-*distinguishable forking* property holds. And let $\mathbf{B} \leftarrow \mathsf{GetRecords}(1^\lambda, \mathsf{st})$ and $\tilde{\mathbf{B}} \leftarrow \mathsf{GetRecords}(1^\lambda, \tilde{\mathsf{st}})$.

Definition 5.2.1 (Evolved Predicate)**.** *An evolved predicate is a polynomial time function* evolved *that takes as input blockchains* $\mathbf{B}$ *and* $\tilde{\mathbf{B}}$

$$\mathsf{evolved}(\mathbf{B}, \tilde{\mathbf{B}}) \in \{0, 1\}$$

It outputs 1 iff $\mathbf{B} = \tilde{\mathbf{B}}$ *or the following holds (i)* $V(\mathbf{B}) = V(\tilde{\mathbf{B}}) = 1$*; (ii)* $\mathbf{B}$ *and* $\tilde{\mathbf{B}}$ *are consistent i.e.* $\mathbf{B}^{\lceil\kappa} \preceq \tilde{\mathbf{B}}$ *where* κ *is the common prefix parameter; (iii) Let* $\ell' = |\tilde{\mathbf{B}}| - |\mathbf{B}|$ *then it holds that* $\ell' \geq \ell_1 + \ell_2$ *and* $\mathsf{u\text{-}stakefrac}(\tilde{\mathbf{B}}, \ell' - \ell_1) > \beta$.

Blockchain Lotteries.

Earlier we mentioned the concept of leader election in PoS-based blockchain protocols. In this kind of lottery any party can win the right to become a slot leader with a probability proportional to its relative stake in the system. Usually, the lottery winner wins the right to propose a new block for the chain, introduce new randomness to the system or become a part of a committee

that carries out some computation. In our encryption scheme we take advantage of this inherent lottery mechanism.

Independent Lotteries. In some applications it is useful to conduct multiple independent lotteries for the same slot sl. Therefore we associate each slot with a set of roles $\mathsf{R}_1, \ldots, \mathsf{R}_n$. Depending on the lottery mechanism, each pair $(\mathsf{sl}, \mathsf{R}_i)$ may yield zero, one or multiple winners. Often, a party can locally compute if it, in fact, is the lottery winner for a given role and the evaluation procedure may equip the party with a proof for others to verify. The below definition details what it means for a party to win a lottery.

Definition 5.2.2 (Lottery Predicate)**.** *A lottery predicate is a polynomial time function* $\mathsf{lottery}$ *that takes as input a blockchain* $\mathbf{B}$*, a slot* sl*, a role* R *and a lottery witness* $\mathsf{sk}_{L,i}$ *and outputs 1 if and only if the party owning* $\mathsf{sk}_{L,i}$ *won the lottery for the role* R *in slot* sl *with respect to the blockchain* $\mathbf{B}$*.*
Formally, we write

$$\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i}) \in \{0,1\}$$

It is natural to establish the set of lottery winning keys $\mathcal{W}_{\mathbf{B},\mathsf{sl},\mathsf{R}}$ for parameters $(\mathbf{B}, \mathsf{sl}, \mathsf{R})$. This is the set of eligible keys satisfying the $\mathsf{lottery}$ predicate.

5.2.2 Commitment Schemes

We refer the reader to Section 3.2.3 for the definition of commitment schemes. In our construction of ECW from cWE (Section 5.5.1), we require our commitments to satisfy an additional property which allows to *extract* message and randomness of a commitment. In particular we assume that our setup outputs both a commitment key and a trapdoor td and that there exists an algorithm Ext such that $\mathsf{Ext}(\mathsf{td}, \mathsf{cm})$ outputs (s, ρ) such that $\mathsf{cm} = \mathsf{Commit}_{\mathsf{ck}}(\mathsf{s}; \rho)$. We remark we can generically obtain this property by attaching to the commitment a NIZK argument of knowledge that shows knowledge of opening, i.e., for the relation $\mathcal{R}^{\text{opn}}(\mathsf{cm}_i; (\mathsf{s}, \rho)) \iff \mathsf{cm}_i = \mathsf{Commit}_{\mathsf{ck}}(\mathsf{s}; \rho)$.

5.2.3 Oblivious Transfer

A 2-round oblivious transfer (OT) protocol between a receiver R and a sender S consists of three polynomial-time algorithms $\Pi_{\mathsf{OT}} = (\Pi^R_{\mathsf{OT}}, \Pi^S_{\mathsf{OT}}, \Pi^O_{\mathsf{OT}})$:

$m^R \leftarrow \Pi^R_{\mathsf{OT}}(b; r^R)$**.** In the first round, the receiver R on input $b \in \{0,1\}$ and random tape $r^R \in \{0,1\}^{\mathsf{poly}(\lambda)(\lambda)}$ generates the OT first message m^R.

$m^S \leftarrow \Pi^S_{\mathsf{OT}}(m^R, (x^0, x^1); r^S)$**.** In the second round, the sender S on input (x^0, x^1), where $x^l \in \{0,1\}^{\mathsf{poly}(\lambda)(\lambda)}$ for $l \in \{0,1\}$, generates the second message m^S using random tape $r^S \in \{0,1\}^{\mathsf{poly}(\lambda)(\lambda)}$.

$x \leftarrow \Pi^O_{\mathsf{OT}}(m^S, b, r^R)$**.** R computes the output $x = \Pi^O_{\mathsf{OT}}(m^S, b, r^R)$.

We require an OT protocol to securely implement the ideal functionality $\mathcal{F}_{\mathsf{OT}}$ given in Fig. 5.2 in the presence of malicious adversaries.

Choose. On input $(\texttt{receive}, \mathsf{sid}, b)$ from R , where $b \in \{0,1\}$, if no messages of the form $(\texttt{receive}, \mathsf{sid}, b)$ is stored, store $(\texttt{receive}, \mathsf{sid}, b)$ and send $(\texttt{receive}, \mathsf{sid})$ to S.

Transfer. On input $(\texttt{send}, \mathsf{sid}, x^0, x^1)$ from S, with $x^0, x^1 \in \{0,1\}^k$, if no messages of the form $(\texttt{send}, \mathsf{sid}, x^0, x^1)$ is stored and a message of the form $(\texttt{receive}, \mathsf{sid}, b)$ is present, send $(\texttt{sent}, \mathsf{sid}, x^b)$ to R.

Figure 5.2: The ideal functionality $\mathcal{F}_{\mathsf{OT}}$ for oblivious transfer

5.2.4 Garbled Circuit

Garbled circuit introduced by [156] is a cryptographic technique that enables two-party secure computation in which two parties do not trust each other and want to jointly evaluate a function over their private inputs. The following definition is from the *garbling schemes* abstraction introduced by Bellare et al. in [21].

Definition 5.2.3 (Garbling Scheme)**.** *Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ be a polynomial-size circuit class. A garbled circuit scheme* GC *for $\mathcal{C}$ consists of four polynomial-time algorithms* $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode})$*:*

$(\mathbf{C}, e, d) \leftarrow \mathsf{Garble}(1^\lambda, C)$**:** *On input a boolean circuit $C \in \mathcal{C}_\lambda$, outputs $(\mathbf{C}, e, d)$, where $\mathbf{C}$ is a garbled circuit, e is encoding information, and d is decoding information.*

$X \leftarrow \mathsf{Encode}(e, x)$**:** *On input e and x, where x is a suitable input for C, outputs a garbled input X.*

$Y = \mathsf{Eval}(\mathbf{C}, X)$**:** *On input $(\mathbf{C}, X)$ as above, outputs a garbled output Y.*

$y \leftarrow \mathsf{Decode}(d, Y)$**:** *On input (d, Y) as above, outputs a plain output y.*

For our construction, we are interested in garbling schemes with the following properties.

Correctness. For any security parameter $\lambda \in \mathbb{N}$, for any circuit $C \in \mathcal{C}_\lambda$, for $(\mathbf{C}, e, d) \leftarrow \mathsf{Garble}(1^\lambda, C)$, and for all suitable input x:

$$\mathsf{Decode}(d, \mathsf{Eval}(\mathbf{C}, \mathsf{Encode}(e, x))) = C(x)$$

Authenticity. For all circuits $C : \{0,1\}^n \to \{0,1\}$, inputs $x \in \{0,1\}^n$, where $n = \mathsf{poly}(\lambda)(\lambda)$, and for all PPT adversaries $\mathcal{A}$,

$$\Pr\left[\begin{matrix}\hat{Y} \neq \mathsf{Eval}(\mathbf{C}, X) \wedge \\ \mathsf{Decode}(d, \hat{Y}) \neq \bot\end{matrix} : \begin{matrix}(\mathbf{C}, e, d) \leftarrow \mathsf{Garble}(1^\lambda, C) \\ X = \mathsf{Encode}(e, x);\ \hat{Y} \leftarrow \mathcal{A}(C, x, \mathbf{C}, X)\end{matrix}\right] \approx_\lambda 0$$

Verifiability. There exists a PPT algorithm Verify such that for all circuits $C : \{0,1\}^{\mathsf{poly}(\lambda)} \to \{0,1\}$,

$$\Pr\left[\mathsf{Verify}(C, \mathbf{C}, e) = 1 \ :\ (\mathbf{C}, e, d) \leftarrow \mathsf{Garble}(1^\lambda, C)\right] = 1$$

5.2.5 (Threshold) Identity Based Encryption

In an IBE scheme, users can encrypt simply with respect to an *identity* (rather than a public key). Given a master secret key, an IBE can generate secret keys that allows to open to specific identities. In our construction of EtF (Section 5.7.1) we rely on a *threshold variant of IBE* (TIBE) where no single party in the system holds the master secret key. Instead, parties in a committee hold a partial master secret key msk_i. Like other threshold protocols, threshold IBE can be generically obtained by "lifting" an IBE through a secret sharing with homomorphic properties (see for example [140]).

We recall the definition of an identity-based encryption (IBE) scheme [37]. An IBE scheme Π_{IBE} consists of the following algorithms:

$\mathsf{Setup}(1^\lambda)$**.** The setup algorithm takes as input a security parameter λ and returns a master key msk together with some publicly known system parameters sp including a master public key mpk, message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$. We assume that all algorithms takes sp as input implicitly.

$\mathsf{IDKeygen}(\mathsf{msk}, \mathsf{ID})$**.** The identity key-generation algorithm takes as input msk and an identity $\mathsf{ID} \in \{0,1\}^*$, and returns a decryption key $\mathsf{sk}_{\mathsf{ID}}$ for ID.

$\mathsf{Enc}(\mathsf{ID}, m)$**.** The encryption algorithm takes as input an identity string $\mathsf{ID} \in \{0,1\}^*$ and $m \in \mathcal{M}$. It returns a ciphertext $ct \in \mathcal{C}$.

$\mathsf{Dec}(ct, \mathsf{sk}_{\mathsf{ID}})$**.** The decryption algorithm takes as input $ct \in \mathcal{C}$ and a decryption key $\mathsf{sk}_{\mathsf{ID}}$. It returns $m \in \mathcal{M}$.

Correctness. An IBE scheme Π_{IBE} should satisfy the standard correctness property, namely for $\mathsf{sk}_{\mathsf{ID}} \leftarrow \mathsf{IDKeygen}(\mathsf{msk}, \mathsf{ID})$ and for any $m \in \mathcal{M}$, we must have:

$$\mathsf{Dec}(\mathsf{Enc}(\mathsf{ID}, m), \mathsf{sk}_{\mathsf{ID}}) = m.$$

where $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$

Security. We use adaptive-identity security [37]. After the challenger runs the setup algorithm, the adversary has access to an oracle $\mathcal{O}_{\mathsf{msk}}$ that on input any id, returns sk_{id}. $\mathcal{A}$ may query the oracle on arbitrary identities of its choice even before selecting the messages m_0, m_1. More formally, we say that Π_{IBE} is secure if any PPT adversary $\mathcal{A}$ has only negligibly greater than 1/2 probability of correctly guessing the bit b in the following game:

1. The challenger runs Setup and outputs sp to $\mathcal{A}$.
2. $\mathcal{A}$ may query the oracle $\mathcal{O}_{\mathsf{msk}}$ that on any input id returns sk_{id}.
3. $\mathcal{A}$ outputs a target identity id^* and two equal-size messages $m_0, m_1 \in \mathcal{M}$.
4. The challenger selects a random bit b and outputs $c^* \leftarrow \mathsf{Enc}(id^*, m_b)$ to $\mathcal{A}$.
5. $\mathcal{A}$ may continue to query $\mathcal{O}_{\mathsf{msk}}$ on any input $id \neq id^*$.
6. $\mathcal{A}$ outputs b'.

where $\mathcal{O}_{\mathsf{msk}}(\mathsf{ID})$ outputs $\mathsf{IDKeygen}(\mathsf{msk}, \mathsf{ID})$.

Threshold IBE.

This is a threshold variant of IBE with the following syntax:

$\Pi_{\mathsf{TIBE}}.\mathsf{Setup}(1^\lambda, n, k) \to (\mathsf{sp}, \mathsf{vk}, \mathbf{msk})$: It outputs some public system parameters sp (including mpk), verification key vk, and vector of master secret key shares $\mathbf{msk} = (\mathsf{msk}_1, \ldots, \mathsf{msk}_n)$ for n with threshold k. We assume that all algorithms takes sp as input implicitly.

$\Pi_{\mathsf{TIBE}}.\mathsf{ShareKG}(i, \mathsf{msk}_i, \mathbf{ID}) \to \theta = (i, \hat{\theta})$: It outputs a private key share $\theta = (i, \hat{\theta})$ for ID given a share of the master secret key.

$\Pi_{\mathsf{TIBE}}.\mathsf{ShareVerify}(\mathsf{vk}, \mathbf{ID}, \theta) \to 0/1$: It takes as input the verification key vk, an identity ID, and a share of master secret key θ, and outputs 0 or 1.

$\Pi_{\mathsf{TIBE}}.\mathsf{Combine}(\mathsf{vk}, \mathbf{ID}, \boldsymbol{\theta}) \to \mathsf{sk}_{\mathbf{ID}}$: It combines the shares $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$ to produce a private key $\mathsf{sk}_{\mathsf{ID}}$ or $\perp$.

$\Pi_{\mathsf{TIBE}}.\mathsf{Enc}(\mathbf{ID}, m) \to \mathsf{ct}$: It encrypts message m for identity ID and outputs a ciphertext ct.

$\Pi_{\mathsf{TIBE}}.\mathsf{Dec}(\mathbf{ID}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ct}) \to m$: It decrypts the ciphertext ct given a private key $\mathsf{sk}_{\mathsf{ID}}$ for identity ID.

Correctness. A TIBE scheme Π_{TIBE} should satisfy two correctness properties:

1. For any identity ID, if $\theta = \Pi_{\mathsf{TIBE}}.\mathsf{ShareKG}(i, \mathsf{msk}_i, \mathsf{ID})$ for $\mathsf{msk}_i \in \mathbf{msk}$, then we have $\Pi_{\mathsf{TIBE}}.\mathsf{ShareVerify}(\mathsf{vk}, \mathsf{ID}, \theta) = 1$.
2. For any ID, if $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_k\}$ where $\theta_i = \Pi_{\mathsf{TIBE}}.\mathsf{ShareKG}(i, \mathsf{msk}_i, \mathsf{ID})$, and $\mathsf{sk}_{\mathsf{ID}} = \Pi_{\mathsf{TIBE}}.\mathsf{Combine}(\mathsf{vk}, \mathsf{ID}, \boldsymbol{\theta})$, then for any $m \in \mathcal{M}$ and $\mathsf{ct} = \Pi_{\mathsf{TIBE}}.\mathsf{Enc}(\mathsf{ID}, m)$ we have $\Pi_{\mathsf{TIBE}}.\mathsf{Dec}(\mathsf{ID}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ct}) = m$.

Constructing TIBE from IBE and Homomorphic Secret Sharing.

Assume a secure $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{IDKeygen}, \mathsf{Enc}, \mathsf{Dec})$. We can transform it into a threshold IBE using homomorphic secret sharing algorithms $(\mathsf{Share}, \mathsf{EvalShare}, \mathsf{Combine})$. A homomorphic secret sharing scheme is a secret sharing scheme with an extra property: given a shared secret, it allows to compute a share of a function of the secret on it. It has the following syntax (which we specialize for the IBE setting):

- $\mathsf{Share}(\mathsf{msk}, k, n) \to (\mathsf{msk}_1, \ldots, \mathsf{msk}_n)$ shares the secret.
- $\mathsf{EvalShare}(\mathsf{msk}_i, f) \to y_i$ obtains a share for $f(\mathsf{msk})$ where f is a function.
- $\mathsf{Combine}((y_i)_{i \in T}) \to y^*$ where T is a set with size above threshold.

We assume all the algorithms above take as input the master public-key for simplicity. The correctness of the homomorphic scheme requires that running $y_i \leftarrow \mathsf{EvalShare}(\mathsf{msk}_i, f)$ on msk_i output of Share and then running $\mathsf{Combine}$ on (a large enough set of) the y_i-s produces the same output as $f(\mathsf{msk})$. We also require that $\mathsf{Combine}$ can reconstruct msk from a large enough set of the msk_i-s.

The construction for threshold IBE is now straightforward:

- at setup time, we produce shares $\mathsf{msk}_1, \ldots, \mathsf{msk}_n$ of the master secret key using the Share algorithm on the master secret key output of Setup.
- encryption is syntactically and functionally the same in both cases.
- to produce a partial secret-key for a certain id, we just run $\mathsf{sk}_i^{\mathsf{ID}} \leftarrow \mathsf{EvalShare}(\mathsf{msk}_i, \mathsf{IBE.IDKeygen}(\mathsf{mpk}, \cdot, \mathsf{ID}))$.
- for decryption, given enough shares for an ID ID, we run on them algorithm $\mathsf{Combine}$ to obtain $\mathsf{sk}_{\mathsf{ID}}$; we then simply run $\mathsf{IBE.Dec}$.

Threshold IBE security. If the homomorphic secret sharing supports up to a threshold k, then we obtain analogous properties for the threshold IBE construction. In particular the threshold IBE satisfies the following simulation properties for any n and threshold k supported by the homomorphic secret sharing scheme[3].

Master secret-key share simulation. For any PPT adversary $\mathcal{A}$ there exists a simulator $\mathsf{Sim}_{\mathsf{msk}}$ such that the following two distributions are indistinguishable.

$$\begin{aligned}
&\{(\mathsf{mpk}, (\mathsf{msk}_i)_{i\in S_{\text{corr}}}) : (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IBE.Setup}(1^\lambda);\\
&\qquad\qquad S_{\text{corr}} \leftarrow \mathcal{A}(\mathsf{mpk}); (\mathsf{msk}_i)_{i\in n} \leftarrow \mathsf{Share}(\mathsf{msk}, n, k)\} \approx\\
&\{(\mathsf{mpk}, (\mathsf{msk}_i)_{i\in S_{\text{corr}}}) : (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IBE.Setup}(1^\lambda);\\
&\qquad\qquad S_{\text{corr}} \leftarrow \mathcal{A}(\mathsf{mpk}); (\mathsf{msk}_i)_{i\in S_{\text{corr}}} \leftarrow \mathsf{Sim}_{\mathsf{msk}}(\mathsf{mpk}, S_{\text{corr}}, n, k)\}
\end{aligned}$$

Key-generation simulation. For any PPT adversary there exists a simulator Sim_{kg} such that the following two distributions are indistinguishable.

$$\begin{aligned}
&\{(\mathsf{mpk}, (\mathsf{sk}_i^{\mathsf{ID}})_{i\in[n]}) : (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IBE.Setup}(1^\lambda);\\
&\qquad\qquad S_{\text{corr}} \leftarrow \mathcal{A}(\mathsf{mpk}); (\mathsf{msk}_i)_{i\in n} \leftarrow \mathsf{Share}(\mathsf{msk}, n, k);\\
&\qquad\qquad \mathsf{ID} \leftarrow \mathcal{A}(\mathsf{mpk}, (\mathsf{msk}_i)_{i\in S_{\text{corr}}});\\
&\qquad\qquad \mathsf{sk}_i^{\mathsf{ID}} \leftarrow \mathsf{EvalShare}(\mathsf{msk}_i, \mathsf{ID}) \text{ for } i \in [n]\} \approx\\
&\{(\mathsf{mpk}, (\mathsf{sk}_i^{\mathsf{ID}})_{i\in[n]}) : (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IBE.Setup}(1^\lambda);\\
&\qquad\qquad S_{\text{corr}} \leftarrow \mathcal{A}(\mathsf{mpk}); (\mathsf{msk}_i)_{i\in n} \leftarrow \mathsf{Share}(\mathsf{msk}, n, k);\\
&\qquad\qquad \mathsf{ID} \leftarrow \mathcal{A}(\mathsf{mpk}, (\mathsf{msk}_i)_{i\in S_{\text{corr}}});\\
&\qquad\qquad (\mathsf{sk}_i^{\mathsf{ID}})_{i\in[n]} \leftarrow \mathsf{Sim}_{\text{kg}}(\mathsf{mpk}, (\mathsf{msk}_i)_{i\in S_{\text{corr}}}, \mathsf{ID})\}
\end{aligned}$$

Robustness of TIBE. We assume a *robust* threshold IBE scheme, where we can verify that each of the ID-specific shares are authenticated, i.e. they have been produced by a party with the related master secret key share. This property can be obtained by assuming an underlying secret sharing scheme which is itself robust. This in turn can be obtained by attaching a NIZK or a homomorphic signature to the share.

[3]The security of this type of construction is proven for example in [140] to which we defer the reader for details.

TIBE with Proactive Secret Sharing. We assume our TIBE to allow for the shares of the master secret keys to be reshared among the committee members which evolve through time. With this goal in mind we can consider a proactive secret sharing scheme which includes a *handover* (each committee member can reshare its share) and *reconstruction* stage (committee members in a new epoch can reconstruct their secret from the output of the handover). We can directly extend a TIBE with such syntax. The resulting scheme should provide the same simulation properties as the ones described above for the non proactive case.

5.3 Modelling EtF

In this section, we present a model for encryption to the future winner of a lottery. In order to argue about a notion of future, we use the blocks of an underlying blockchain ledger and their relative positions in the chain to specify points in time. Intuitively, our notion allows for creating ciphertexts that can only be decrypted by a party that is selected to perform a certain role R at a future slot sl according to a lottery scheme associated with a blockchain protocol. The winner of the lottery at a point in the future with respect to a blockchain state $\tilde{\mathbf{B}}$ is determined by the lottery predicate defined in Section 5.2.1, *i.e.* the winner is the holder of a lottery secret key sk such that $\mathsf{lottery}(\tilde{\mathbf{B}}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}) = 1$. However, notice that the winner might only be determined by a blockchain state produced in the future as a result of the blockchain protocol execution. This makes it necessary for the ciphertext to encode an initial state $\mathbf{B}$ of the blockchain that allows for verifying that a future state $\tilde{\mathbf{B}}$ (presented at the time of decryption) has indeed been produced as a result of correct protocol execution. This requirement is captured by the evolving blockchain predicate defined in Section 5.2.1, *i.e.* $\mathsf{evolved}(\mathbf{B}, \tilde{\mathbf{B}}) = 1$ iff $\tilde{\mathbf{B}}$ is obtained as a future state of executing the blockchain protocol departing from $\mathbf{B}$.

Definition 5.3.1 (Encryption to the Future)**.** *A pair of PPT algorithms $\mathcal{E} = (\mathsf{Enc}, \mathsf{Dec})$ in the the context of a blockchain Γ^V is an EtF-scheme with evolved predicate* evolved *and a lottery predicate* lottery*. The algorithms work as follows.*

Encryption. $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, m)$ *takes as input an initial blockchain $\mathbf{B}$, a slot* sl*, a role* R *and a message m. It outputs a ciphertext* ct *- an encryption to the future.*

Decryption. $m/\perp \leftarrow \mathsf{Dec}(\tilde{\mathbf{B}}, \mathsf{ct}, \mathsf{sk})$ *takes as input a blockchain state $\tilde{\mathbf{B}}$, a ciphertext* ct *and a secret key* sk *and outputs the original message m or $\perp$.*

An EtF must satisfy the following properties:

Correctness. An EtF-scheme is said to be correct if for honest parties i and j, there exists a negligible function μ such that for all $\mathsf{sk}, \mathsf{sl}, \mathsf{R}, m$:

$$\left| \Pr \left[\begin{array}{c} \mathsf{view} \leftarrow \mathsf{EXEC}^{\Gamma}(\mathcal{A}, \mathcal{Z}, 1^{\lambda}) \\ \mathbf{B} = \mathsf{GetRecords}(\mathsf{view}_i) \\ \tilde{\mathbf{B}} = \mathsf{GetRecords}(\mathsf{view}_j) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, m) \\ \mathsf{evolved}(\mathbf{B}, \tilde{\mathbf{B}}) = 1 \end{array} : \begin{array}{c} \mathsf{lottery}(\tilde{\mathbf{B}}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}) = 0 \\ \vee\ \mathsf{Dec}(\tilde{\mathbf{B}}, \mathsf{ct}, \mathsf{sk}) = m \end{array} \right] - 1 \right| \leq \mu(\lambda)$$

Security. We establish a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. In Section 5.2.1 we describe how $\mathcal{A}$ and $\mathcal{Z}$ execute a blockchain protocol. In addition, we now let the

$\mathsf{view}^r \leftarrow \mathsf{EXEC}^{\Gamma}_r(\mathcal{A}, \mathcal{Z}, 1^\lambda)$ ▷ $\mathcal{A}$ executes Γ with $\mathcal{Z}$ until round r
$(\mathbf{B}, \mathsf{sl}, \mathsf{R}, m_0, m_1) \leftarrow \mathcal{A}(\mathsf{view}^r_{\mathcal{A}})$ ▷ $\mathcal{A}$ outputs challenge parameters
$b \leftarrow\$ \{0,1\}$
$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, m_b)$
$\mathsf{st} \leftarrow \mathcal{A}(\mathsf{view}^r_{\mathcal{A}}, \mathsf{ct})$ ▷ $\mathcal{A}$ receives challenge ct
$\mathsf{view}^{\tilde{r}} \leftarrow \mathsf{EXEC}^{\Gamma}_{(\mathsf{view}^r, \tilde{r})}(\mathcal{A}, \mathcal{Z}, 1^\lambda)$ ▷ Execute from view^r until round $\tilde{r}$
$(\tilde{\mathbf{B}}, b') \leftarrow \mathcal{A}(\mathsf{view}^{\tilde{r}}_{\mathcal{A}}, \mathsf{st})$
if $\mathsf{evolved}(\mathbf{B}, \tilde{\mathbf{B}}) = 1$ **then** ▷ $\tilde{\mathbf{B}}$ is a valid evolution of $\mathbf{B}$
  if $sk^{\mathcal{A}}_{L,j} \notin \mathcal{W}_{\tilde{\mathbf{B}},\mathsf{sl},\mathsf{R}}$ **then** ▷ $\mathcal{A}$ does not win role R
    return $b \oplus b'$
  end if
end if
return $\hat{b} \leftarrow\$ \{0,1\}$

Figure 5.3: $\mathsf{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}}$

adversary interact with the challenger in a game $\mathsf{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}}$ described in Fig. 5.3. The game can be summarized as follows:

1. $\mathcal{A}$ executes the blockchain protocol Γ together with $\mathcal{Z}$ and at some round r chooses a blockchain $\mathbf{B}$, a role R for the slot sl and two messages m_0 and m_1 and sends it all to $\mathcal{C}$.
2. $\mathcal{C}$ chooses a random bit b and encrypts the message m_b with the parameters it received and sends ct to $\mathcal{A}$.
3. $\mathcal{A}$ continues to execute the blockchain until some round $\tilde{r}$ where the blockchain $\tilde{\mathbf{B}}$ is obtained and $\mathcal{A}$ outputs a bit b'.

If the adversary is a lottery winner for the challenge role R in slot sl, the game outputs a random bit. If the adversary is not a lottery winner for the challenge role R in slot sl, the game outputs $b \oplus b'$. The reason for outputting a random guess in the game when the challenge role is corrupted is as follows. Normally the output of the IND-CPA game is $b \oplus b'$ and we require it to be 1 with probability 1/2. This models that the guess b' is independent of b. This, of course, cannot be the case when the challenge role is corrupted. We therefore output a random guess in these cases. After this, any bias of the output away from 1/2 still comes from b' being dependent on b.

Definition 5.3.2 (IND-CPA Secure EtF)**.** *An EtF-scheme $\mathcal{E} = (\mathsf{Enc}, \mathsf{Dec})$ in the context of a blockchain protocol Γ executed by PPT machines $\mathcal{A}$ and $\mathcal{Z}$ is said to be* IND-CPA *secure if, for any $\mathcal{A}$ and $\mathcal{Z}$, there exists a negligible function μ such that for $\lambda \in \mathbb{N}$:*

$$\left|2 \cdot \Pr\left[\mathsf{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}} = 1\right] - 1\right| \leq \mu(\lambda)$$

5.3.1 ECW as a Special Case of EtF

In this section we focus on a special class of EtF. We call schemes in this class ECW schemes. ECW is particularly interesting since the underlying lottery is always conducted with respect to the current blockchain state. This has the following consequences

1. $\mathbf{B} = \tilde{\mathbf{B}}$ means that $\mathsf{evolved}(\mathbf{B}, \tilde{\mathbf{B}}) = 1$ is trivially true.
2. The winner of role R in slot sl is already defined in $\mathbf{B}$.

It is easy to see that this kind of EtF scheme is simpler to realize since there is no need for checking if the blockchain has "correctly" evolved. Furthermore, all lottery parameters like stake distribution and randomness extracted from the blockchain are static. Thus, an adversary has no way to move stake between accounts in order to increase its chance of winning the lottery. Note that, when using an ECW scheme, the lottery winner is already decided at encryption time. In other words, there is no delay and the moment a ciphertext is produced the receiver is chosen.

5.4 Witness Encryption over Commitments (cWE)

Here, we describe witness encryption over commitments that is a relaxed notion of witness encryption. In witness encryption parties encrypt to a public input for some NP statement. In cWE we have two phases: first parties provide a (honestly generated) commitment cm of their private input s. Later, anybody can encrypt to a public input for an NP statement which *also* guarantees correct opening of the commitment. Importantly, in applications, the first message in our model can be reused for many different invocations.

Remark 5.4.1 (Comparing cWE and WE). We observe that cWE is weaker than standard WE because of its deterministic flavor. In standard WE we encrypt without having any "pointer" to an alleged witness, but in cWE it requires the witness to be implicitly *known* at encryption time through the commitment (to which it is bound). That is why—as for the weak flavors of witness encryption in [29]—we believe it would be misleading to just talk about WE. This is true in particular since we show cWE can be constructed from standard assumptions such as oblivious transfer and garbled circuits (Section 5.4.2), whereas constructions of WE from standard assumptions are still an open problem or require strong primitives like indistinguishability obfuscation. Finally we stress a difference with the trivial "interactive" WE proposed in [98] (Section 1.3): cWE is still non-interactive after producing a once-and-for-all reusable commitment.

5.4.1 Definition

The type of relations we consider are of the following form: a statement $\mathsf{x} = (\mathsf{cm}, C, y)$ and a witness $\mathsf{w} = (\mathsf{s}, \rho)$ are in the relation (i.e., $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$) iff "$\mathsf{cm}$ commits to some secret value s using randomness ρ, and $C(\mathsf{s}) = y$". Here, C is a circuit in some circuit class $\mathcal{C}$ and y is the expected output of the function.
Formally, we define witness encryption over commitments as follows:

Definition 5.4.2 (Witness encryption over commitments)**.** *Let* $\mathbf{Com} = (\mathsf{Setup}, \mathit{Commit})$ *be a non-interactive commitment scheme. A cWE-scheme for witness encryption over commitments with circuit class* $\mathcal{C}$ *and commitment scheme* $\mathbf{Com}$ *consists of a pair of algorithms* $\Pi_{\mathsf{cWE}} = (\mathsf{Enc}, \mathsf{Dec})$*:*

Encryption phase. $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ck}, \mathsf{x}, m)$ *on input a commitment key* ck*, a statement* $\mathsf{x} = (\mathsf{cm}, C, y)$ *such that* $C \in \mathcal{C}$*, and a message* $m \in \{0,1\}^*$*, generates a ciphertext* ct*.*

Decryption phase. $m/\perp \leftarrow \mathsf{Dec}(\mathsf{ck}, \mathsf{ct}, \mathsf{w})$ *on input a commitment key* ck*, a ciphertext* ct*, and a witness* w*, returns a message* m *or* $\perp$*.*

A cWE should satisfy *correctness* and *semantic security* as defined below.

(Perfect) Correctness. An honest prover with a statement $\mathsf{x} = (\mathsf{cm}, C, y)$ and witness $\mathsf{w} = (\mathsf{s}, \rho)$ such that $\mathsf{cm} = \mathsf{Commit}_{\mathsf{ck}}(\mathsf{s}; \rho)$ and $C(\mathsf{s}) = y$ can always decrypt with overwhelming probability. More precisely, a cWE with circuit class $\mathcal{C}$ and commitment scheme $\mathbf{Com}$ has perfect correctness if for all $\lambda \in \mathbb{N}$, $C \in \mathcal{C}$, $\mathsf{ck} \in \mathsf{Range}(\mathbf{Com}.\mathsf{Setup})$, $\mathsf{s} \in \mathcal{S}_m$, randomness $\rho \in \mathcal{S}_r$, commitment $\mathsf{cm} \leftarrow \mathbf{Com}.\mathsf{Commit}_{\mathsf{ck}}(\mathsf{s}; \rho)$, and bit message $m \in \{0,1\}^*$, it holds that

$$\Pr\left[\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ck}, (\mathsf{cm}, C, C(\mathsf{s})), m); m' \leftarrow \mathsf{Dec}(\mathsf{ck}, \mathsf{ct}, (\mathsf{s}, \rho)) \ : \ m = m'\right] = 1$$

(Weak) Semantic Security. Intuitively, encrypting with respect to a false statement (with honest commitment) produces indistinguishable ciphertexts. Formally, there exists a negligible function μ such that for all $\lambda \in \mathbb{N}$, all auxiliary strings aux and all PPT adversaries $\mathcal{A}$:

$$\left| 2 \cdot \Pr\left[\begin{array}{r} \mathsf{ck} \leftarrow \mathbf{Com}.\mathsf{Setup}(1^\lambda) \\ (\mathsf{st}, \mathsf{s}, \rho, C, y, m_0, m_1) \leftarrow \mathcal{A}(\mathsf{ck}, \mathsf{aux}) \\ \mathsf{cm} \leftarrow \mathbf{Com}.\mathsf{Commit}_{\mathsf{ck}}(\mathsf{s}; \rho); b \leftarrow\$ \{0,1\} \ : \ \mathcal{A}(\mathsf{st}, \mathsf{ct}) = b \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ck}, (\mathsf{cm}, C, y), m_b) \\ \mathsf{ct} := \perp \text{ if } C(\mathsf{s}) = y,\ C \notin \mathcal{C} \text{ or } |m_0| \neq |m_1| \end{array}\right] - 1 \right| \leq \mu(\lambda)$$

To show the construction of ECW from cWE, we need a stronger notion of semantic security where the adversary additionally gets to see ciphertexts of the challenge message under true statements with unknown to $\mathcal{A}$ witnesses. Below we formalize this property and show that weak semantic security together with hiding of the commitment imply strong semantic security.

Strong Semantic Security.

Informally, this property states that encrypting a message m with respect to a false statement $\mathsf{x} = (\mathsf{cm}, C, y)$ produces indistinguishable ciphertexts to an adversary $\mathcal{A}$ who knows the commitment opening, even if $\mathcal{A}$ gets to see encryptions of m under other (possibly true) statements $\mathsf{x}_i = (\mathsf{cm}_i, C, y)$ but with unknown commitment opening. Formally, there exists a negligible function μ such that for all $\lambda \in \mathbb{N}$, all auxiliary strings aux and all PPT adversaries $\mathcal{A}$:

$$\left| 2 \cdot \Pr\left[\begin{array}{r} \mathsf{ck} \leftarrow \mathbf{Com}.\mathsf{Setup}(1^\lambda) \\ (\mathsf{st}, \mathsf{s}, \rho, C, y, m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{com}}(\cdot)}(\mathsf{ck}, \mathsf{aux}) \\ \mathsf{cm} \leftarrow \mathbf{Com}.\mathsf{Commit}_{\mathsf{ck}}(\mathsf{s}; \rho); b \leftarrow\$ \{0,1\} \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ck}, \mathsf{x} = (\mathsf{cm}, C, y), m_b) \ : \ \mathcal{A}(\mathsf{st}, \mathbf{ct}) = b \\ \forall \mathsf{cm}_i \in \mathcal{Q} : \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ck}, \mathsf{x}_i = (\mathsf{cm}_i, C, y), m_b) \\ \mathbf{ct} := \{\mathsf{ct}\} \cup \{\mathsf{ct}_i\}_{i \in [|\mathcal{Q}|]} \\ \mathbf{ct} := \perp \text{ if } C(\mathsf{s}) = y \text{ or } C \notin \mathcal{C} \end{array}\right] - 1 \right| \leq \mu(\lambda)$$

where $\mathcal{O}_{\text{com}}(\cdot)$ is a commitment oracle parametrized by ck and defined as follows: on input s_i, computes and returns $\mathsf{cm}_i \leftarrow \mathbf{Com}.\mathsf{Commit}_{\mathsf{ck}}(\mathsf{s}_i; \rho_i)$ for some randomness ρ_i, and stores cm_i in $\mathcal{Q}$.

Lemma 5.4.3. *Let* $\mathbf{Com} = (\mathsf{Setup}, \mathit{Commit})$ *be a non-interactive commitment scheme. Let* Π_{cWE} *be a witness encryption over commitments for some circuit class* $\mathcal{C}$ *over commitment scheme* $\mathbf{Com}$*. If* Π_{cWE} *has weak semantic security, and* $\mathbf{Com}$ *has hiding property, then* Π_{cWE} *has strong semantic security.*

Proof. Assume $\mathcal{A}$ is a PPT adversary against strong semantic security. We construct an efficient adversary $\mathcal{B}$ that breaks weak semantic security of Π_{cWE} with non-negligible advantage.

First, $\mathcal{B}$ runs $\mathcal{A}$ with the commitment key ck received from the challenger. $\mathcal{B}$ then simulates the oracle $\mathcal{O}_{\text{com}}$ for $\mathcal{A}$ in the natural way. Namely, for any input s_i, it outputs $\mathsf{cm}_i \leftarrow \mathbf{Com}.\mathsf{Commit}_{\mathsf{ck}}(\mathsf{s}_i; \rho_i)$ for some randomness $\rho_i \in \mathcal{S}_r$, and stores cm_i in $\mathcal{Q}$. Upon receiving a tuple $(\mathsf{st}, \mathsf{s}, \rho, C, y, m_0, m_1)$ from $\mathcal{A}$, $\mathcal{B}$ forwards the tuple to the challenger. Upon receiving the challenge ciphertext ct (for the encryption of m_b) from the challenger, $\mathcal{B}$ generates a ciphertext ct_i for each commitment $\mathsf{cm}_i \in \mathcal{Q}$. To do so, $\mathcal{B}$ selects $c \leftarrow\!\!\$\ \{0,1\}$ and computes $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ck}, (\mathsf{cm}_i, C, y), m_c)$ for any $\mathsf{cm}_i \in \mathcal{Q}$. Next, $\mathcal{B}$ checks whether $C(\mathsf{s}) \neq y$, and if so forwards $\mathbf{ct} := \{\mathsf{ct}\} \cup \{\mathsf{ct}_i\}_{i \in [|\mathcal{Q}|]}$ to $\mathcal{A}$. Otherwise, $\mathcal{B}$ outputs a random guess $b' \leftarrow\!\!\$\ \{0,1\}$ for the bit b. Finally, upon receiving a guess b' from $\mathcal{A}$, $\mathcal{B}$ forwards b' to the challenger. It is easy to see that if $c = b$, then $\mathcal{B}$ is perfectly simulating strong semantic security game for $\mathcal{A}$.

To prove the lemma, we define $|\mathcal{Q}| + 2$ hybrid distributions such that the first hybrid corresponds to the strong semantic security and the last hybrid corresponds to the above game simulated by $\mathcal{B}$. We conclude the proof by showing that an adversary with non-negligible advantage in the first hybrid implies the existence of an efficient adversary with non-negligible advantage in the last hybrid.

Hybrid 0. This is the strong semantic security game. Namely,

1. The adversary $\mathcal{A}$ receives the commitment key ck, where $\mathsf{ck} \leftarrow \mathbf{Com}.\mathsf{Setup}(1^\lambda)$.
2. $\mathcal{A}$ adaptively makes commitment queries for messages s_i, and for each receives $\mathsf{cm}_i \leftarrow \mathbf{Com}.\mathsf{Commit}_{\mathsf{ck}}(\mathsf{s}_i; \rho_i)$.
3. After some number of queries listed in $\mathcal{Q}$, $\mathcal{A}$ outputs a tuple $(\mathsf{st}, \mathsf{s}, \rho, C, y, m_0, m_1)$ for which $C(\mathsf{s}) \neq y$. The challenger samples a random bit $b \leftarrow\!\!\$\ \{0,1\}$, generates encryptions of m_b via $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ck}, (\mathsf{cm}, C, y), m_b)$, and $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ck}, (\mathsf{cm}_i, C, y), m_b)$ for all $\mathsf{cm}_i \in \mathcal{Q}$, and sends $\mathbf{ct} := \{\mathsf{ct}\} \cup \{\mathsf{ct}_i\}_{i \in [|\mathcal{Q}|]}$ to $\mathcal{A}$ as the challenge ciphertext.
4. Eventually, $\mathcal{A}$ outputs a guess b' for the bit b.

Hybrids $k = 1, \ldots, |\mathcal{Q}|$. Same as the previous hybrid, except the first k ciphertexts $\{\mathsf{ct}_i\}_{i \in [k]}$ are computed with respect to cm_i being a commitment of s. Namely,

1. *Identical to Hybrid 0.*
2. *Identical to Hybrid 0.*
3. The challenger samples a random bit $b \leftarrow\!\!\$\ \{0,1\}$, generates encryptions of m_b via $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ck}, (\mathsf{cm}, C, y), m_b)$, and $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ck}, (\mathsf{cm}_i, C, y), m_b)$ $(i = 1, \ldots, |\mathcal{Q}|)$ computed as before, except in the first k ciphertexts $\{\mathsf{ct}_i\}_{i \in [k]}$, the commitment cm_i is computed as $\mathsf{cm}_i \leftarrow \mathbf{Com}.\mathsf{Commit}_{\mathsf{ck}}(\mathsf{s}; \rho_i)$ for some randomness $\rho_i \in \mathcal{S}_r$.

4. *Identical to Hybrid 0.*

Hybrid $k = |\mathcal{Q}| + 1$. Same as the previous hybrid, except the ciphertexts $\{\mathsf{ct}_i\}_{i\in[|\mathcal{Q}|]}$ are encryptions of m_c for a uniformly random $c \leftarrow\!\!\$\ \{0,1\}$. Namely,

1. *Identical to Hybrid $|\mathcal{Q}|$.*
2. *Identical to Hybrid $|\mathcal{Q}|$.*
3. The challenger samples random bits $b \leftarrow\!\!\$\ \{0,1\}$ and $c \leftarrow\!\!\$\ \{0,1\}$, and generates encryptions of m_b and m_c respectively via $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ck}, (\mathsf{cm}, C, y), m_b)$, and $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ck}, (\mathsf{cm}_i, C, y), m_c)$ $(i = 1, \ldots, |\mathcal{Q}|)$.
4. *Identical to Hybrid $|\mathcal{Q}|$.*

For $k = 0, \ldots, |\mathcal{Q}| + 1$, denote by adv_i the advantage of $\mathcal{A}$ in guessing the bit b in Hybrid i. It is easy to see that for any $0 \leq i < |\mathcal{Q}|$, we have $|\mathsf{adv}_i - \mathsf{adv}_{i+1}| \leq \mathsf{negl}(\lambda)(\lambda)$, where $\mathsf{negl}(\lambda)(\lambda)$ is some negligible function. This is because the distributions $\mathcal{D}_i$ and $\mathcal{D}_{i+1}$ respectively defined by the hybrids i and $i + 1$ only differ in their $(i + 1)$-th ciphertext, which is the encryption of m_b under a commitment of s_{i+1} for $\mathcal{D}_i$ and encryption of m_b under a commitment of s for $\mathcal{D}_{i+1}$. Thus, the difference $|\mathsf{adv}_i - \mathsf{adv}_{i+1}|$ is exactly equal to the adversary's advantage in the hiding experiment. By the hiding property of the commitment scheme, it thus follows that this difference is negligible.
Furthermore, observe that in the last hybrid, $c = b$ with probability $1/2$ and hence we have that with probability at least $1/2$, the two distributions $\mathcal{D}_{|\mathcal{Q}|}$ and $\mathcal{D}_{|\mathcal{Q}|+1}$ are identical. This, together with the fact that $\mathcal{D}_0$ and $\mathcal{D}_{|\mathcal{Q}|}$ have a negligible difference imply that having an efficient adversary with non-negligible advantage ε against hybrid 0 results in a non-negligible advantage $\varepsilon/2$ against hybrid $|\mathcal{Q}| + 1$. This completes the proof of the lemma. □

5.4.2 Constructions of cWE

From Multi-Sender 2P-NISC [8].

A cWE scheme can be constructed from protocols for Multi-Sender (reusable) Non-Interactive Secure Computation (MS-NISC) [8]. In such protocols, there is a receiver R with input x who first broadcasts an encoding of its input, and then later every sender S_i with input y_i can send a single message to R that conveys only $f(x, y_i)$. This is done while preserving privacy of inputs and correctness of output. The ideal functionality of MS-NISC as presented in [8] is depicted in Fig. 5.4.

In Fig. 5.5, we show how to construct cWE by having black-box access to $\mathcal{F}_{\mathsf{MS\text{-}NISC}}$. The main idea is that a party acts as a receiver and sends the first message in MS-NISC containing its witness w in order to provide a "commitment" to that witness. Later on, any other party can use this "commitment" to create a cWE ciphertext by sending an encryption of the message and acting as the sender of the MS-NISC to provide a second message that allows for evaluating a function $f(\mathsf{w}, y)$ that outputs a decryption key iff the witness w satisfies a given relation. Note that the ideal functionalities used in the construction are stated for clarity and is not compatible with our game-based notion of security for cWE. By assuming a concrete secure realization of the above functionalities, one can argue about security using the corresponding simulator and use that to extract witnesses from commitments and make the proof go through.

Assume $f(\perp, \cdot) = f(\cdot, \perp) = \perp$.

- Initialize a list, L, of pairs of strings.
- Upon receiving a message $(\texttt{input}, x)$ from R, store x and continue.
  1. Upon receiving message $(\texttt{input}, y)$ from S_i, insert the pair (S_i, y) into L. If R is corrupted send $(S_i, f(x, y))$ to the adversary. Otherwise, send $(\texttt{messageReceived}, S_i)$ to R.
  2. Upon receiving a message `getOutputs` from R, send $\{(S_i, f(x, y))\}_{(S_i,y)\in L}$ to R.

Figure 5.4: MS-NISC Functionality $\mathcal{F}_{\mathsf{MS\text{-}NISC}}$

Initialization: Initialize $\mathcal{F}_{\mathsf{MS\text{-}NISC}}$ by instantiating a list L of pairs of strings.

Commit: R proceeds as follows:

- Commits to its witness w by calling $\mathcal{F}_{\mathsf{MS\text{-}NISC}}$ on input $(\texttt{input}, \mathsf{w})$.

Encryption: S proceeds as follows:

- Generates a key k of length $|m|$ and encrypts the message m as $\mathsf{ct} \leftarrow k \oplus m$.
- Calls $\mathcal{F}_{\mathsf{MS\text{-}NISC}}$ on input (x, k) and sends ct directly to R.

Decryption: R receives $(\texttt{messageReceived}, S)$ from $\mathcal{F}_{\mathsf{MS\text{-}NISC}}$ and ct from S and proceeds as follows:

- Calls $\mathcal{F}_{\mathsf{MS\text{-}NISC}}$ on input `getOutputs`.
- Upon receiving k from $\mathcal{F}_{\mathsf{MS\text{-}NISC}}$, outputs $m \leftarrow k \oplus \mathsf{ct}$.

Figure 5.5: Construction of cWE based on MS-NISC

We observe that the above construction actually yields a stronger notion of cWE where the statement x is private which is not a requirement in our setting. This asymmetry between sender and receiver privacy was also observed by others [122] and it opens the door for efficient constructions using oblivious transfer (OT) and privacy-free garbled circuits as described in [157].

From Garbled Circuits and Oblivious Transfer.

Instead of relying on the full MS-NISC functionality in a black-box way, we now do a careful analysis resulting in a protocol which uses only the properties of MS-NISC needed to obtain a protocol that satisfies the definition of cWE.

We observe that the correctness property in the definition of cWE only requires that a correctly generated ciphertext can be decrypted by the decryption algorithm. Thus, we expect the second message of MS-NISC functionality to be generated correctly. In particular, when looking into the internals of the protocol in [8], we observe that we can construct cWE from a MS-NISC protocol without the precautions against a malicious sender S. However, we still want to make sure that we preserve authenticity of the underlying garbled circuit scheme. This property guarantees that no garbled output can be constructed different from what is dictated by the function and its inputs. In other words, the only thing a malicious receiver can do with the garbled circuit is Evaluate it on the committed input. Finally, we observe that privacy of input is not a requirement for the sender. Thus, we can consider variants of garbled circuit schemes without privacy guarantees.

Privacy-free Garbled Circuits. One of the most efficient GC schemes in terms of communication is the scheme by [157] based on a technique called half-gates. Using their technique in the privacy-free setting results in garbled circuits containing one ciphertext for each AND gate and no ciphertexts for XOR gates.

cWE from privacy-free GC and OT. We now present an efficient construction of cWE using only a privacy-free garbled circuit and oblivious transfer.

Let $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode}, \mathsf{Verify})$ be a garbled circuit with correctness and authenticity, and $\Pi_{\mathsf{OT}} = (\Pi^R_{\mathsf{OT}}, \Pi^S_{\mathsf{OT}}, \Pi^O_{\mathsf{OT}})$ be an oblivious transfer protocol that realizes $\mathcal{F}_{\mathsf{OT}}$. We consider two parties E and D that respectively play the role of the encryptor and the decryptor in an execution of the cWE scheme. The construction of $\Pi_{\mathsf{cWE}} = (\mathsf{Enc}, \mathsf{Dec})$ with commitment Π^R_{OT} for circuit class $\mathcal{C}$ is given in Fig. 5.6.

Theorem 5.4.4. *Let $\mathcal{C}$ be a class of circuits. Let Π_{OT} be an OT protocol that realizes $\mathcal{F}_{\mathsf{OT}}$ and GC be a correct and authentic garbling scheme. The cWE scheme Π_{cWE} for $\mathcal{C}$ in Fig. 5.6 is correct and semantically secure as defined in Definition 5.4.2.*

Proof. **(Correctness).** Follows directly from the correctness property of the Π_{OT} and GC.
(Semantic Security). Assume that $\mathcal{A}$ is a PPT adversary against semantic security of Π_{cWE} such that, for adversarially chosen values $(\mathsf{s}, \rho, C, y, m_0, m_1)$, given an encryption of m_b under statement $\mathsf{x} = (\mathsf{cm}, C, y)$, where $\mathsf{cm} = \mathsf{Commit}(\mathsf{s}; \rho)$ and $y \neq C(\mathsf{s})$, $\mathcal{A}$ can guess the bit b with non-negligible advantage. We first observe that by the construction of Π_{cWE}, $\mathcal{A}$ can guess b correctly only by distinguishing the correct label k^1 from random. Informally, given that $C(\mathsf{s}) \neq y$, there are only two possible cases in which $\mathcal{A}$ can distinguish k^1 from random: either by the ability to gain knowledge about invalid labels $k_j^{1-\mathsf{s}_j}$ that do not correspond to $\mathcal{A}$'s committed value, or by the ability to gain knowledge about k^1 directly. We show that a successful adversary in the first case can be used to break the sender security of Π_{OT} whereas a successful adversary in the second case can be exploited to break the the authenticity of GC.

In order to formally prove semantic security, we first define the experiment $\mathsf{Exp}^{\mathsf{SS\text{-}b}}_{\mathcal{A},\lambda}$ in Fig. 5.7. We define b' as the output of $\mathsf{Exp}^{\mathsf{SS\text{-}b}}_{\mathcal{A},\lambda}$. Note that $\mathsf{Exp}^{\mathsf{SS\text{-}b}}_{\mathcal{A},\lambda}$ corresponds to the semantic security experiment of Π_{cWE} in Definition 5.4.2, except that b is fixed.

To prove the theorem, let us assume by contradiction that there is an adversary $\mathcal{A}$ that breaks the semantic security of Π_{cWE}. That is, for a non-negligible function ϵ, we have

$$\left|\Pr[1 \leftarrow \mathsf{Exp}^{\mathsf{SS\text{-}0}}_{\mathcal{A},\lambda}] - \Pr[1 \leftarrow \mathsf{Exp}^{\mathsf{SS\text{-}1}}_{\mathcal{A},\lambda}]\right| \geq \epsilon(\lambda)$$

Primitives: A correct and authentic garbling scheme $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode})$, and a 2-round OT $\Pi_{\mathsf{OT}} = (\Pi^R_{\mathsf{OT}}, \Pi^S_{\mathsf{OT}}, \Pi^O_{\mathsf{OT}})$.
Commit: D with secret $\mathsf{s} \in \{0,1\}^n$ plays the role of the receiver in n instances of Π_{OT} and computes $(\mathsf{cm}, \mathsf{w})$ as follows:

- Select $\rho^R_j \leftarrow\$ \{0,1\}^\lambda$, and compute $m^R_j \leftarrow \Pi^R_{\mathsf{OT}}(\mathsf{s}_j; \rho^R_j)$ for $j \in [n]$.
- Define $\mathsf{cm} = \{m^R_j\}_{j\in[n]}$, and $\mathsf{w} = (\mathsf{s}, \{\rho^R_j\}_{j\in[n]})$. Note that w can be seen as an opening of cm.

Common inputs: A security parameter λ, a circuit $C \in \mathcal{C}$, a commitment key ck, and a statement $\mathsf{x} = (\mathsf{cm}, C, y)$.
Encryption.: E plays the role of the sender in n instances of Π_{OT} and computes a ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$ as follows:

1. Let C_{x} be a circuit that realizes the following relation $\mathcal{R}$ on x: $\mathcal{R}(\mathsf{x} = (\mathsf{cm}, C, y), (\mathsf{s}, \mathbf{d})) = 1$ iff $(\mathsf{s}, \mathbf{d})$ opens cm and $C(\mathsf{s}) = y$. Compute $(\mathbf{C}, e, d) \leftarrow \mathsf{Garble}(1^\lambda, C_{\mathsf{x}})$, where $e := \{k^0_j, k^1_j\}_{j\in[n]}$, and $d := (k^0, k^1) \in \{0,1\}^{2|m|}$.
2. For $j \in [n]$, select $\rho^S_j \leftarrow\$ \{0,1\}^\lambda$, and compute $m^S_j = \Pi^S_{\mathsf{OT}}(k^0_j, k^1_j, m^R_j; \rho^S_j)$.
3. Compute $\mathsf{ct}_1 = k^1 \oplus m$ and $\mathsf{ct}_2 = (\mathbf{C}, \{m^S_j\}_{j\in[n]})$.
4. Send $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$ to D.

Decryption: Given $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$ and $\mathsf{w} = (\mathsf{s}, \{\rho^R_j\}_{j\in[n]})$, D proceeds as follows:

1. Parse ct_2 as $(\mathbf{C}, \{m^S_j\}_{j\in[n]})$.
2. Execute $k^{\mathsf{s}_j}_j = \Pi^O_{\mathsf{OT}}(m^S_j, \mathsf{s}_j, \rho^R_j)$ for $j \in [n]$, and $Y = \mathsf{Eval}(\mathbf{C}, \{k^{\mathsf{s}_j}_j\}_{j\in[n]})$.
3. Compute $m = Y \oplus \mathsf{ct}_1$.

Figure 5.6: cWE based on GC and OT

$(\mathsf{st}, \mathsf{s}, \rho, C, y, m_0, m_1) \leftarrow \mathcal{A}(1^\lambda)$
$\rho = \rho_1 || \ldots || \rho_n$; $\mathsf{cm} = \{m^R_j\}_{j\in[n]}$, where $m^R_j \leftarrow \Pi^R_{\mathsf{OT}}(\mathsf{s}_j; \rho_j)\ \forall j \in [n]$.
$\mathsf{x} := (\mathsf{cm}, C, y)$; $(\mathbf{C}, e, d) \leftarrow \mathsf{Garble}(1^\lambda, C_{\mathsf{x}})$ where $e := \{k^0_j, k^1_j\}_{j\in[n]}$, and $d := (k^0, k^1)$.
$\rho^S_j \leftarrow\$ \{0,1\}^\lambda$; $m^S_j = \Pi^S_{\mathsf{OT}}(k^0_j, k^1_j, m^R_j; \rho^S_j)$ $(\forall j \in [n])$.
$\mathsf{ct}_1 = k^1 \oplus m_b$ and $\mathsf{ct}_2 = (\mathbf{C}, \{m^S_j\}_{j\in[n]})$; $\mathsf{ct} := (\mathsf{ct}_1, \mathsf{ct}_2)$.
$\mathsf{ct} := \perp$ if $C(\mathsf{s}) = y$ or $C \notin \mathcal{C}$ or $|m_0| \neq |m_1|$
$b' \leftarrow \mathcal{A}(\mathsf{st}, \mathsf{ct})$

Figure 5.7: $\mathsf{Exp}^{\mathsf{SS\text{-}b}}_{\mathcal{A},\lambda}$

We now use a standard hybrid argument and define several games, where the first is $\mathsf{Exp}^{\mathsf{SS\text{-}0}}_{\mathcal{A},\lambda}$, the last is $\mathsf{Exp}^{\mathsf{SS\text{-}1}}_{\mathcal{A},\lambda}$, and the intermediate hybrids are defined as follows:
Hybrid 0 is defined as $\mathsf{Exp}^{\mathsf{SS\text{-}0}}_{\mathcal{A},\lambda}$.
Hybrid 1 is the same as **Hybrid 0**, except that the messages $\{m_j^S\}_{j\in[n]}$ are computed by the OT simulator i.e., as $\{m_j^S\}_{j\in[n]} \leftarrow \mathsf{Sim}(1^\lambda, \{m_j^R\}_{j\in[n]})$.
Hybrid 2 is the same as **Hybrid 1**, except that ct_1 is defined as $\mathsf{ct}_1 := k^1 \oplus m_1$.
Hybrid 3 is defined as $\mathsf{Exp}^{\mathsf{SS\text{-}1}}_{\mathcal{A},\lambda}$.

By assumption, $\mathcal{A}$ must distinguish some pair of adjacent intermediate hybrids. That is, for some $i \in \{0, 1, 2\}$, we must have

$$\left|\Pr[1 \leftarrow \mathsf{Hybrid}^i_{\mathcal{A},\lambda} - \Pr[1 \leftarrow \mathsf{Hybrid}^{i+1}_{\mathcal{A},\lambda}]\right| \geq \frac{1}{3}\epsilon(\lambda)$$

We now analyze all three cases:

- $i = 0$. Notice that the only difference between **Hybrid 0** and **Hybrid 1** is that in the former, the sender's message $\{m_j^S\}_{j\in[n]}$ is computed by a real sender (World 0), whereas in the latter, it is computed by the simulator (World 1). Assuming that $\mathcal{A}$ can distinguish hybrids 0 and 1, we construct an adversary $\mathcal{B}$ against sender security of Π_{OT} that distinguishes the two worlds with the same probability. $\mathcal{B}$ works as follows:

  1. $\mathcal{B}$ invokes $\mathcal{A}(1^\lambda)$ and obtains $(\mathsf{s}, \rho, C, y, m_0, m_1)$.
  2. If $|m_0| \neq |m_1|$ or $C(\mathsf{s}) = y$, $\mathcal{B}$ aborts; otherwise, it parses $\rho = \rho_1||\ldots||\rho_n$ and defines $\mathsf{cm} = \{m_j^R\}_{j\in[n]}$, where $m_j^R \leftarrow \Pi^R_{\mathsf{OT}}(\mathsf{s}_j; \rho_j)$ for $j \in [n]$. Let $\mathsf{x} = (\mathsf{cm}, C, y)$. As an environment controlling the OT execution, $\mathcal{B}$ provides the input of sender and receiver to the OT challenger as follows:
     - computes a garbling of circuit C_{x} (as defined in Fig. 5.6) by $(\mathbf{C}, e, d) \leftarrow \mathsf{Garble}(1^\lambda, C_{\mathsf{x}})$ and sends the input keys to the OT challenger as the sender's input.
     - sends s to the OT challenger as the receiver's choice bits.
  3. The OT challenger computes the sender's message $\{m_j^S\}_{j\in[n]}$ either by invoking a real sender (World 0), or by invoking the simulator (World 1), and sends it to $\mathcal{B}$.
  4. $\mathcal{B}$ parses $d = (k^0, k^1)$, and forwards $\mathsf{ct} := (\mathsf{ct}_1, \mathsf{ct}_2)$ to the cWE adversary $\mathcal{A}$, where $\mathsf{ct}_1 = k^1 \oplus m_0$ and $\mathsf{ct}_2 = (\mathbf{C}, \{m_j^S\}_{j\in[n]})$.

  It is clear that $\mathcal{B}$ has the same advantage in breaking sender security of Π_{OT} as $\mathcal{A}$ in distinguishing the two hybrids.

- $i = 1$. The only difference in **Hybrid 1** and **Hybrid 2** is in how we generate ct_1 (that is $\mathsf{ct}_1 := k^1 \oplus m_{i-1}$ in **Hybrid i**). To argue indistinguishability of the two hybrids, it suffices to show that k^1 is indistinguishable from random. To achieve this, we observe that because in both hybrids, the sender's message $\{m_j^S\}_{j\in[n]}$ is computed by the simulator i.e., as $\{m_j^S\}_{j\in[n]} \leftarrow \mathsf{Sim}(1^\lambda, \{m_j^R\}_{j\in[n]})$, $\mathcal{A}$ cannot distinguish k^1 from random by the ability of knowing invalid labels. Thus, the only way $\mathcal{A}$ can distinguish k^1 from random should be by directly forging an output key k^1 for the garbled circuit $\mathbf{C}$. It is therefore straightforward to use a successful adversary that distinguishes the two hybrids with non-negligible advantage to break the authenticity of the underlying garbling scheme.

- $i = 2$. This is handled identically to $i = 0$, except that in this case $\mathsf{ct}_1 := k^1 \oplus m_1$ encrypts m_1 instead of m_0.

We conclude the proof by this observation that in any of the three cases, we reach a contradiction and thus our assumption of the existence of $\mathcal{A}$ against the semantic security of Π_{cWE} cannot be true. □

Remark 5.4.5. The commitment scheme in Π_{cWE} is the receiver's algorithm of Π_{OT} and therefore by UC-security of Π_{OT}, it satisfies both extractability and hiding property. Using Lemma 5.4.3 and weak semantic security shown in Theorem 5.4.4, one can then conclude that Π_{cWE} also achieves strong semantic security.

5.5 Construction of ECW

Here we show a novel construction of ECW from cWE. We then show alternative constructions through instantiations from previous work.

5.5.1 ECW from cWE

In this section we realize the notion of ECW from cWE. We define our scheme with respect to a set of parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ executing a blockchain protocol Γ as described in Section 5.2.1, *i.e.* each party P_i has access to the blockchain ledger and is associated to a tuple $(\mathsf{Sig.pk}_i, \mathsf{aux}_i, \mathsf{st}_i)$ registered in the genesis block for which it has corresponding secret keys $(\mathsf{Sig.sk}_i, \mathsf{sk}_{L,i})$. Our construction uses as a main building block a witness encryption scheme over commitments $\Pi_{\mathsf{cWE}} = (\mathsf{Enc}_{\mathsf{cWE}}, \mathsf{Dec}_{\mathsf{cWE}})$; we assume the commitments to be extractable. The class of circuits $\mathcal{C}$ of Π_{cWE} includes the lottery predicate $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i})$. We let each party publish an initial commitment of its witness. This way we can do without any interaction for encryption/decryption through a one-time setup where parties publish the commitments over which all following encryptions are done. We construct our ECW scheme Π_{ECW} as follows:

System Parameters: We assume that a commitment key $\mathsf{Setup}(1^\lambda) \to \mathsf{ck}$ is contained in the genesis block B_0 of the underlying blockchain.

Setup Phase: All parties $P_i \in \mathcal{P}$ proceed as follows:

1. Compute a commitment $\mathsf{cm}_i \leftarrow \mathsf{Commit}_{\mathsf{ck}}(\mathsf{sk}_{L,i}; \rho_i)$ to $\mathsf{sk}_{L,i}$ using randomness ρ_i. We abuse the notation and define P_i's secret key as $\mathsf{sk}_{L,i}||\rho_i$.
2. Compute a signature $\sigma_i \leftarrow \mathsf{Sig}_{\mathsf{Sig.sk}_i}(\mathsf{cm}_i)$.
3. Publish $(\mathsf{cm}_i, \sigma_i)$ on the blockchain by executing $\mathsf{Broadcast}(1^\lambda, (\mathsf{cm}_i, \sigma_i))$.

Encryption $\mathsf{Enc}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, m)$**:** Construct a circuit C that encodes the predicate $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i})$, where $\mathbf{B}$, sl and R are hardcoded and $\mathsf{sk}_{L,i}$ is the witness. Let $\mathcal{P}_{Setup}$ be the set of parties with non-zero relative stake and a valid setup message $(\mathsf{cm}_i, \sigma_i)$ published in the common prefix $\mathbf{B}^{\lceil\kappa}$ (if P_i has published more than one valid $(\mathsf{cm}_i, \sigma_i)$, only the latest one is considered). For every $P_i \in \mathcal{P}_{Setup}$, compute $\mathsf{ct}_i \leftarrow \mathsf{Enc}_{\mathsf{cWE}}(\mathsf{ck}, \mathsf{x}_i = (\mathsf{cm}_i, C, 1), m)$. Output $\mathsf{ct} = \big(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \{\mathsf{ct}_i\}_{P_i \in \mathcal{P}_{Setup}}\big)$.

Decryption $\mathsf{Dec}(\mathbf{B}, \mathsf{ct}, \mathsf{sk})$**:** For a given $\mathsf{sk} := \mathsf{sk}_{L,i}||\rho_i$ such that $\mathsf{cm}_i = \mathsf{Commit}_{\mathsf{ck}}(\mathsf{sk}_{L,i}; \rho_i)$ and $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i}) = 1$ (for parameters $\mathbf{B}, \mathsf{sl}, \mathsf{R}$ from ct), output $m \leftarrow \mathsf{Dec}_{\mathsf{cWE}}(\mathsf{ck}, \mathsf{ct}_i, (\mathsf{sk}_{L,i}, \rho_i))$. Otherwise, output $\perp$.

Theorem 5.5.1. *Let* $\mathbf{Com} = (\mathsf{Setup}, \mathsf{Commit})$ *be a non-interactive extractable commitment scheme and* $\Pi_{\mathsf{cWE}} = (\mathsf{Enc}_{\mathsf{cWE}}, \mathsf{Dec}_{\mathsf{cWE}})$ *be a strong semantically secure cWE over* $\mathbf{Com}$ *for a circuit class* $\mathcal{C}$ *encoding the lottery predicate* $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i})$ *as defined in Section 5.4. Let* Γ *be a blockchain protocol as defined in Section 5.2.1.* Π_{ECW} *is an* IND-CPA*-secure ECW scheme as per Definition 5.3.2.*

Proof. Assume by contradiction that there exists an adversary $\mathcal{A}_{\text{ECW}}$ with non-negligible advantage in $\text{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}}$ in the ECW setting as described in Section 5.3.1. We construct an adversary $\mathcal{A}_{\mathsf{cWE}}$ with black-box access to $\mathcal{A}_{\text{ECW}}$ that has non-negligible advantage in breaking strong semantic security of Π_{cWE} as defined above. We assume (w.l.o.g.) that $\mathcal{A}_{\text{ECW}}$ only corrupts one party P_a[4]. $\mathcal{A}_{\mathsf{cWE}}$ proceeds as follows:

1. Upon receiving the commitment key ck from the challenge, $\mathcal{A}_{\mathsf{cWE}}$ proceeds as follows:

   a) $\mathcal{A}_{\mathsf{cWE}}$ acts as the environment $\mathcal{Z}$ orchestrating the execution of the blockchain protocol Γ towards $\mathcal{A}_{\text{ECW}}$, placing the commitment key ck in the genesis block. $\mathcal{A}_{\mathsf{cWE}}$ acts exactly as $\mathcal{Z}$ in $\text{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}}$.

   b) $\mathcal{A}_{\mathsf{cWE}}$ simulates honest parties P_h executing the setup phase and publishing a valid $(\mathsf{cm}_h, \sigma_h)$ on the blockchain. To simulate cm_h for each honest party, $\mathcal{A}_{\mathsf{cWE}}$ calls the oracle $\mathcal{O}_{\text{com}}$ on some random input $\mathsf{sk}_{L,h}$ and sets cm_h to be $\mathcal{O}_{\text{com}}$'s output.

   c) At some point, $\mathcal{A}_{\text{ECW}}$ outputs challenge parameters $\mathbf{B}, \mathsf{sl}, \mathsf{R}, m_0, m_1$ from its view of the blockchain. $\mathcal{A}_{\mathsf{cWE}}$ constructs a circuit C that encodes the predicate $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i})$, where $\mathbf{B}$, sl and R are hardcoded and $\mathsf{sk}_{L,i}$ is the witness.

   d) Finally, if there exists a valid setup message $(\mathsf{cm}_a, \sigma_a)$ published in the common prefix $\mathbf{B}^{\lceil\kappa}$ by P_a (*i.e.* the corrupted party P_a is in $\mathcal{P}_{Setup}$), $\mathcal{A}_{\mathsf{cWE}}$ extracts $\mathsf{sk}_{L,a}, \rho_a$ from cm_a using the extractability of the commitment scheme $\mathbf{Com}$ and outputs $(\mathsf{state}, \mathsf{sk}_{L,a}, \rho_a, C, 1, m_0, m_1)$ to the challenger. Otherwise, $\mathcal{A}_{\mathsf{cWE}}$ outputs $(\mathsf{state}, \mathsf{sk}_{L,k}, \rho_k, C, 1, m_0, m_1)$ to the challenger, where $\mathsf{sk}_{L,k}, \rho_k$ are chosen at random and such that $C(\mathsf{sk}_{L,k}) \neq 1$.

2. Upon receiving ciphertexts $\mathbf{ct} = \{\mathsf{ct}\} \cup \{\mathsf{ct}_h\}_{P_h \in \mathcal{P}_{Setup}}$ from the challenger, if $P_a \in \mathcal{P}_{Setup}$, then $\mathsf{ct} = \mathsf{ct}_a$ was computed w.r.t. P_a's commitment cm_a and ct_h computed w.r.t. the honest party's commitment cm_h. Otherwise, if only honest parties are in $\mathcal{P}_{Setup}$, $\mathcal{A}_{\mathsf{cWE}}$ forwards the ECW ciphertexts $\mathbf{ct} = \{\mathsf{ct}_h\}_{P_h \in \mathcal{P}_{Setup}}$ to $\mathcal{A}_{\text{ECW}}$. $\mathcal{A}_{\mathsf{cWE}}$ continues the execution of Γ with $\mathcal{A}_{\text{ECW}}$ from the round where it stopped when $\mathcal{A}_{\text{ECW}}$ outputted challenge parameters $\mathbf{B}, \mathsf{sl}, \mathsf{R}, m_0, m_1$.

3. Upon receiving a guess b' from $\mathcal{A}_{\text{ECW}}$, $\mathcal{A}_{\mathsf{cWE}}$ forwards b' to the challenger.

First, notice that $\mathcal{A}_{\text{ECW}}$ has the same access to the underlying blockchain protocol Γ (and to the system parameters in the genesis block) as in $\text{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}}$. In case $\mathcal{A}_{\text{ECW}}$ provided a valid

[4] In reality there will be more than one corrupted party; the main argument underlying our proof holds regardless.

setup message, it receives $\mathbf{ct}$ containing a cWE ciphertext ct_a generated with respect to its commitment cm_a and the circuit encoding the lottery predicate $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i})$, where $\mathbf{B}$, sl and R provided by $\mathcal{A}_{\mathrm{ECW}}$ are hardwired. Moreover, $\mathbf{ct}$ contains ciphertexts ct_h for each cm_h, encrypting the same m_b as in ct_a. Hence, $\mathbf{ct}$ is distributed exactly as in $\mathrm{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}}$. If $\mathcal{A}_{\mathrm{ECW}}$ has non-negligible advantage in $\mathrm{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}}$, it is able to distinguish whether ct_a contains m_0 or m_1 with non-negligible advantage even though it does not have $\mathsf{sk}_{L,a}$ and $\mathsf{cm}_a \leftarrow \mathsf{Commit}_{\mathsf{ck}}(\mathsf{sk}_{L,a}; \rho_a)$ such that $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,a}) = 1$, *i.e.* it does not have $\mathsf{sk}_{L,a}$ such that $C(\mathsf{sk}_{L,a}) = 1$. This means that, by forwarding guess b' from $\mathcal{A}_{\mathrm{ECW}}$, $\mathcal{A}_{\mathsf{cWE}}$ in the cWE semantic security game has the same advantage as $\mathcal{A}_{\mathrm{ECW}}$ in $\mathrm{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}}$. In case it did not provide a valid setup message, $\mathcal{A}_{\mathrm{ECW}}$ only sees $\mathbf{ct} = \{\mathsf{ct}_h\}_{P_h \in \mathcal{P}_{Setup}}$ with ct_h being an encryption of m_b with respect to the commitments cm_h for which it does not know the opening. Hence, $\mathbf{ct}$ is again distributed exactly as in $\mathrm{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}}$ with probability 1. In this case, by an analogous argument as before, the advantage of the adversary $\mathcal{A}_{\mathsf{cWE}}$ must be the same as the advantage of the adversary $\mathcal{A}_{\mathrm{ECW}}$ in $\mathrm{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}}$.

Since we assume that $\mathcal{A}_{\mathrm{ECW}}$ has a non-negligible advantage, $\mathcal{A}_{\mathsf{cWE}}$ will also obtain a non-negligible advantage and thus break the cWE scheme we assume is secure. Hence, Π_{ECW} is an IND-CPA-secure ECW scheme. □

5.5.2 Other Instantiations

ECW from target anonymous channels [33, 103]. As mentioned before, another approach to construct ECW can be based on a recent line of work that aims to design secure-MPC protocols where parties should remain anonymous until they speak [33, 102, 103]. The baseline of these results is to establish a communication channel to random parties, while preserving their anonymity. It is quite clear that such anonymous channels can be used to realize our definition of ECW for the underlying lottery predicate that defines to whom the anonymous channel is established. Namely, to encrypt m to a role R at a slot sl with respect to a blockchain state $\mathbf{B}$, create a target anonymous channel to $(\mathsf{R}, \mathsf{sl})$ over $\mathbf{B}$ by using the above approaches and send m via this channel. Depending on the lottery predicate that specifies which random party the channel is created for, a recipient with the secret key who wins this lottery can retrieve m. To include some concrete examples, the work of Benhamouda et al. [33] proposed the idea of using a "nomination" process, where a nominating committee chooses a number of random parties $\mathcal{P}$, look up their public keys, and publish a re-randomization of their key. This allows everyone to send messages to $\mathcal{P}$ while keeping their anonymity. The work of [33] answered this question differently by delegating the nomination task to the previous committees without requiring a nominating committee. That is, the previous committee runs a secure-MPC protocol to choose a random subset of public keys, and broadcasts the rerandomization of the keys. To have a MPC protocol that scales well with the total number of parties, they define a new flavour of private information retrieval (PIR) called random-index PIR (or RPIR) and show how each committee—playing the role of the RPIR client—can select the next committee with the complexity only proportional to the size of the committee. There are two constructions of RPIR proposed in [103], one based on Mix-Nets and the other based on FHE. Since the purpose of the constructions described is to establish a target-anonymous channel to a random party, one can consider them as examples of a stronger notion of ECW with anonymity and a specific lottery predicate that selects *a single* random party from the entire population as the winner.

ECW from [76]. Derler and Slamanig [76] (DS) constructed a variant of WE for a restricted

class of algebraic languages. In particular, a user can conduct a Groth-Sahai (GS) proof for the satisfiability of some pairing-product equations (PPEs). Such a proof contains commitments to the witness using randomness only known by this user. The proof can be used by anyone to encrypt a message resulting in a ciphertext which can only be decrypted by knowing this randomness. More formally, they consider a type of WE associated with a proof system $\Pi = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ consisting of two rounds. In the first round, a recipient computes and broadcasts $\pi \leftarrow \mathsf{P}(\mathsf{crs}, \mathsf{x}, \mathsf{w})$. Later, a user can verify the proof and encrypt a message m under (x, π) if $\mathsf{V}(\mathsf{crs}, \mathsf{x}, \pi) = 1$. We note that the proof π does not betray the user conducting the proof and therefore it can use an anonymous broadcast channel to communicate the proof to the encrypting party in order to obtain anonymous ECW. Moreover, although GS proofs may look to support only a restricted class of statements based on PPEs, they are expressive enough to cover all the statements arising in pairing-based cryptography. This indicates the applicability of this construction for any VRF-based lottery where the VRF is algebraic and encodable as a set of PPEs. This interactive ECW just described yields an improvement in communication complexity at the cost of having an extra round of interaction.

From Signatures of Knowledge. Besides the above instantiations, we point out a (potentially more inefficient) abstract construction from zero-knowledge signatures of knowledge (SoK) [59] (roughly, a non-malleable non-interactive zero-knowledge proof). This is similar in spirit to the previous instantiation and can be seen as a generalization. Assume each party has a (potentially ephemeral) public key. At the time the lottery winner has been decided, the winners can post a SoK showing knowledge of the secret key corresponding to their pk *and* that their key is a winner of the lottery. To encrypt, one would first verify the SoK and then encrypt with respect to the corresponding public key.

5.6 YOSO Multiparty Computation from ECW

In this section we show how ECW can be used as the crucial ingredient in setting up a YOSO MPC. So far we have only focused on IND-CPA secure ECW, which falls short of role assignment in the sense of [102]. In general role assignment requires the following properties which are not provided by ECW (or EtF):

1. Multiple parties must be able to send messages to the same role (in most applications this requires IND-CCA).

2. Parties must authenticate messages on behalf of a role they executed in the past (authentication from the past)

3. A party assigned to a given role must stay covert until the role is executed.

We will define a number of properties needed for EtF to realize applications such as role assignment. We start by looking at CCA security for an EtF scheme. We then introduce the notion of Authentication from the Past (AfP) and definition of unforgeability and privacy guarantees. Finally, we introduce the notion of YOSO-friendly blockchains that have inbuilt lotteries with properties that are needed to conduct YOSO MPC and corresponding EtF and AfP schemes.

$\mathsf{view}^r \leftarrow \mathsf{EXEC}_r^{\Gamma}(\mathcal{A}^{\mathcal{O}_{\mathsf{EtF}}}, \mathcal{Z}, 1^\lambda)$ ▷ $\mathcal{A}$ executes Γ with $\mathcal{Z}$ until round r
$(\mathbf{B}, \mathsf{sl}, \mathsf{R}, m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{EtF}}}(\mathsf{view}^r_{\mathcal{A}})$ ▷ $\mathcal{A}$ outputs challenge parameters
$b \leftarrow\$ \{0,1\}$
$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, m_b)$
$\mathsf{st} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{EtF}}}(\mathsf{view}^r_{\mathcal{A}}, \mathsf{ct})$ ▷ $\mathcal{A}$ receives challenge ct
$\mathsf{view}^{\tilde{r}} \leftarrow \mathsf{EXEC}^{\Gamma}_{(\mathsf{view}^r, \tilde{r})}(\mathcal{A}^{\mathcal{O}_{\mathsf{EtF}}}, \mathcal{Z}, 1^\lambda)$ ▷ Execute from view^r until round $\tilde{r}$
$(\tilde{\mathbf{B}}, b') \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{EtF}}}(\mathsf{view}^{\tilde{r}}_{\mathcal{A}}, \mathsf{st})$
if $\mathsf{evolved}(\mathbf{B}, \tilde{\mathbf{B}}) = 1$ **then** ▷ $\tilde{\mathbf{B}}$ is a valid evolution of $\mathbf{B}$
  if $\mathsf{sk}^{\mathcal{A}}_{L,j} \notin \mathcal{W}_{\tilde{\mathbf{B}},\mathsf{R},\mathsf{sl}} \wedge \mathsf{ct} \notin \mathcal{Q}_{\mathsf{EtF}}$ **then** ▷ $\mathcal{A}$ does not win role R
    return $b \oplus b'$
  end if
end if
return $g \leftarrow\$ \{0,1\}$

Figure 5.8: $\mathsf{Game}^{\mathsf{IND\text{-}CCA2}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathsf{E}}$

5.6.1 IND-CCA EtF

In this section we define what it means for an EtF to be IND-CCA secure. This security property is useful in many applications where more encryptions are done towards the same slot and role. As in the definition of IND-CPA, we establish a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. We introduce a decryption oracle, $\mathcal{O}_{\mathsf{EtF}}$, which on input ct returns the decryption of ciphertext. Furthermore, the $\mathcal{O}_{\mathsf{EtF}}$ maintains a list of ciphertext queries $\mathcal{Q}_{\mathsf{EtF}}$. Fig. 5.8 shows the details of the game.

Definition 5.6.1 (IND-CCA2 Secure EtF)**.** *Formally, an EtF-scheme $\mathcal{E}$ is said to be IND-CCA2 secure in the context of a blockchain protocol Γ executed by PPT machines $\mathcal{A}$ and $\mathcal{Z}$ if there exists a negligible function μ such that for $\lambda \in \mathbb{N}$:*

$$\left|2 \cdot \Pr\left[\mathsf{Game}^{\mathsf{IND\text{-}CCA2}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}} = 1\right] - 1\right| \leq \mu(\lambda)$$

To add IND-CCA2 security to an IND-CPA secure EtF scheme (as defined in Definition 5.3.2) we can use standard transformations such as [90, 150]. In the transformation based on [150] we could add to the setup of the blockchain a CRS for a simulation-sound extractable NIZK. When encrypting m to a role R the sender will send along a proof of knowledge of the plaintext m. We get the challenge ciphertext from the IND-CPA game and use the ZK property to simulate the NIZK proof. We can use the extraction trapdoor of the proof system to simulate the CCA decryption oracles by simulation soundness. When the IND-CCA2 adversary makes a guess, we make the same guess. The details of the construction and proof follow using standard techniques and are omitted. On the other hand, the popular transformation of [90] allows for simulating CCA decryption oracles by observing the adversary's queries to a random oracle, which should not be an issue since an EtF scheme is likely already running on top of a blockchain which is secure in the random oracle model. We leave the construction of concretely efficient IND-CCA2 EtF as future work.

5.6.2 Authentication from the Past (AfP)

When the winner of a role R_1 sends a message m to a future role R_2 then it is typically also needed that R_2 can be sure that the message m came from a party P which, indeed, won the role R_1. Most PoS blockchains deployed in practice have a lottery where a certificate can be released proving that P won the role R_1. In order to formalize this concept, we introduce an AfP scheme with a corresponding EUF-CMA game representing the authentication property.

Definition 5.6.2 (Authentication from the Past)**.** *A pair of PPT algorithms $\mathcal{U} = (\mathsf{Sign}, \mathsf{Verify})$ is a scheme for authenticating messages as a winner of a lottery in the past in the context of blockchain Γ with lottery predicate* $\mathsf{lottery}$.

Authenticate. $\sigma \leftarrow \mathsf{AfP.Sign}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}, m)$ *takes as input a blockchain $\mathbf{B}$, a slot* sl*, a role* R*, a secret key* sk*, and a message m. It outputs a signature σ that authenticates the message m.*

Verify. $\{0,1\} \leftarrow \mathsf{AfP.Verify}(\tilde{\mathbf{B}}, \mathsf{sl}, \mathsf{R}, \sigma, m)$ *uses the blockchain $\tilde{\mathbf{B}}$ to ensure that σ is a signature on m produced by the secret key winning the lottery for slot* sl *and role* R.

Furthermore, an AfP-scheme has the following properties:

Correctness. *An AfP-scheme is said to be correct if for honest parties i and j, there exists a negligible function μ such that for all* $\mathsf{sk}, \mathsf{sl}, \mathsf{R}, m$*:*

$$\left| \Pr \left[\begin{array}{c} \mathsf{view} \leftarrow \mathit{EXEC}^{\Gamma}(\mathcal{A}, \mathcal{Z}, 1^{\lambda}) \\ \mathbf{B} = \mathsf{GetRecords}(\mathsf{view}_i) \\ \tilde{\mathbf{B}} = \mathsf{GetRecords}(\mathsf{view}_j) \\ \sigma \leftarrow \mathsf{AfP.Sign}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}, m) \end{array} : \begin{array}{c} \mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}) = 0 \\ \vee\ \mathsf{lottery}(\tilde{\mathbf{B}}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}) = 0 \\ \vee\ \mathsf{AfP.Verify}(\tilde{\mathbf{B}}, \mathsf{sl}, \mathsf{R}, \sigma, m) = 1 \end{array} \right] - 1 \right| \leq \mu(\lambda)$$

In other words, an AfP on a message from an honest party with a view of the blockchain $\mathbf{B}$ can attest to the fact that the sender won the role R *in slot* sl*. If another party, with blockchain $\tilde{\mathbf{B}}$ agrees, then the verification algorithm will output 1.*

Security. *We here describe the game detailed in Fig. 5.9 representing the security of an AfP scheme. The algorithm represents a standard EUF-CMA game where the adversary has access to a signing oracle $\mathcal{O}_{\mathsf{AfP}}$ which it can query with a slot* sl*, a role* R *and a message m_i and obtain AfP signatures $\sigma_i = \mathsf{AfP.Sign}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_j, m_i)$ where $\mathsf{sk}_j \in \mathcal{W}_{\mathbf{B},\mathsf{sl},\mathsf{R}}$ i.e. $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_j) = 1$. The oracle maintains the list of queries $\mathcal{Q}_{\mathsf{AfP}}$.*
Formally, an AfP-scheme $\mathcal{U}$ is said to be EUF-CMA secure in the context of a blockchain protocol Γ executed by PPT machines $\mathcal{A}$ and $\mathcal{Z}$ if there exists a negligible function μ such that for $\lambda \in \mathbb{N}$:

$$\Pr\left[\mathrm{Game}^{\mathsf{EUF\text{-}CMA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{U}} = 1\right] \leq \mu(\lambda)$$

General AfP.

In general we can add authentication to a message as follows. Recall that P_i wins R if $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i}) = 1$. Here, $\mathcal{R}(\mathsf{x} = (\mathbf{B}, \mathsf{sl}, \mathsf{R}), \mathsf{w}) = \mathsf{lottery}(\mathsf{x}, \mathsf{w})$ is an NP relation where all parties know x but only the winner knows a witness w such that $\mathcal{R}(\mathsf{x}, \mathsf{w}) = 1$. We can

$\mathsf{view} \leftarrow \mathsf{EXEC}^{\Gamma}(\mathcal{A}, \mathcal{Z}, 1^{\lambda})$ ▷ $\mathcal{A}$ executes Γ with $\mathcal{Z}$
$(\mathbf{B}, \mathsf{sl}, \mathsf{R}, m', \sigma') \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{AfP}}}(\mathsf{view}_{\mathcal{A}})$
if $(m' \in \mathcal{Q}_{\mathsf{AfP}}) \vee (\mathsf{sk}^{\mathcal{A}}_{L,j} \in \mathcal{W}_{\mathbf{B},\mathsf{sl},\mathsf{R}})$ **then** ▷ $\mathcal{A}^{\mathcal{O}_{\mathsf{AfP}}}$ won or queried illegal m'
  return 0
end if
$\mathsf{view}^{\tilde{r}} \leftarrow \mathsf{EXEC}^{\Gamma}_{(\mathsf{view}^{r},\tilde{r})}(\mathcal{A}, \mathcal{Z}, 1^{\lambda})$ ▷ Execute from view^{r} until round $\tilde{r}$
$\tilde{\mathbf{B}} \leftarrow \mathsf{GetRecords}(\mathsf{view}^{\tilde{r}}_{i})$
if $\mathsf{evolved}(\mathbf{B}, \tilde{\mathbf{B}}) = 1$ **then**
  if $\mathsf{AfP.Verify}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \sigma', m') = 1$ **then** ▷ $\mathcal{A}$ successfully forged an AfP
    return 1
  end if
end if
return 0

Figure 5.9: $\mathsf{Game}^{\mathsf{EUF\text{-}CMA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{U}}$

therefore use a signature of knowledge (SoK) [59] to sign m under the knowledge of $\mathsf{sk}_{L,i}$ such that $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i}) = 1$. This will attest that the message m was sent by a winner of the lottery for R. In Section 5.6.4, we show more efficient construction of AfP by exploring the structure of PoS-based blockchains with VRF lotteries.

5.6.3 AfP Privacy

Just EUF-CMA security is not sufficient for an AfP mechanism to be YOSO friendly. It must also preserve the privacy guarantees of the lottery predicate, guaranteeing that the adversary does not gain any undue advantage in predicting when a party is selected to perform a role after it uses AfP to authenticate a message. To appreciate this fact, we consider the case where instead of creating a signature of knowledge of $\mathsf{sk}_{L,i}$ on message m we simply use a regular EUF-CMA secure signature scheme to sign the message concatenated with $\mathsf{sk}_{L,i}$, revealing the signature public key, the resulting signature and $\mathsf{sk}_{L,i}$ itself as a means of authentication. By definition, this will still constitute an existentially unforgeable AfP but will also reveal whether the party who owns $\mathsf{sk}_{L,i}$ is the winner when future lotteries are conducted. The specific privacy property we seek is that an adversary, observing AfP tags from honest parties, cannot use this information to enhance its chances in predicting the winners of lotteries for roles for which an AfP tag has not been published. On the other hand, the identity of a party who won the lottery for a given role is not kept private when it publishes an AfP tag on behalf of this role, which is not an issue in a YOSO-setting since corruption after-the-fact is futile. Specifically, we allow an AfP tag to be linked to the identity of the party who generated it. Note, that this kind of privacy is different from notions like *k*-anonymity since the success of the adversary in guessing lottery winners with high accuracy depends on the stake distribution. The stake distribution is public in most PoS-settings and, thus, a privacy definition must take into account this inherent leakage.

Definition 5.6.3 (AfP Privacy.)**.** *An AfP scheme $\mathcal{U}$ with corresponding lottery predicate* lottery *is private if a PPT adversary $\mathcal{A}$ is unable to distinguish between the scenarios defined in the*

following two algorithms with more than negligible probability in the security parameter.

Scenario 0 ($b = 0$)**.** *In this scenario, $\mathcal{A}$ is first running the blockchain Γ together with the environment $\mathcal{Z}$. At round r, $\mathcal{A}$ is allowed to interact with the oracle $\mathcal{O}_{\mathsf{AfP}}$ (see Definition 5.6.2). The adversary then continues the execution until round $\tilde{r}$ where it outputs a bit b'.*

Scenario 1 ($b = 1$)**.** *This scenario is identical to scenario 0 but instead of interacting with $\mathcal{O}_{\mathsf{AfP}}$, the adversary interacts with a simulator* Sim.

$b = 0$
$\mathsf{view}^r \leftarrow \mathsf{EXEC}^{\Gamma}_{r}(\mathcal{A}, \mathcal{Z}, 1^\lambda)$
$\mathcal{A}^{\mathcal{O}_{\mathsf{AfP}}}(\mathsf{view}^r_{\mathcal{A}})$
$\mathsf{view}^{\tilde{r}} \leftarrow \mathsf{EXEC}^{\Gamma}_{(\mathsf{view}^r, \tilde{r})}(\mathcal{A}, \mathcal{Z}, 1^\lambda)$
return $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{AfP}}}(\mathsf{view}^{\tilde{r}}_{\mathcal{A}})$

$b = 1$
$\mathsf{view}^r \leftarrow \mathsf{EXEC}^{\Gamma}_{r}(\mathcal{A}, \mathcal{Z}, 1^\lambda)$
$\mathcal{A}^{\mathsf{Sim}}(\mathsf{view}^r_{\mathcal{A}})$
$\mathsf{view}^{\tilde{r}} \leftarrow \mathsf{EXEC}^{\Gamma}_{(\mathsf{view}^r, \tilde{r})}(\mathcal{A}, \mathcal{Z}, 1^\lambda)$
return $b' \leftarrow \mathcal{A}^{\mathsf{Sim}}(\mathsf{view}^{\tilde{r}}_{\mathcal{A}})$

We let $\mathrm{Game}^{\mathsf{AfP\text{-}PRIV}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{U}}$ *denote the game where a coin-flip decides whether the adversary is executed in scenario 0 or scenario 1. We say that the adversary wins the game (i.e.* $\mathrm{Game}^{\mathsf{AfP\text{-}PRIV}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{U}} = 1$*) iff $b' = b$. Finally, an AfP scheme $\mathcal{U}$ is called private in the context of the blockchain Γ executed together with environment $\mathcal{Z}$ if the following holds for a negligible function μ.*

$$\left|2 \cdot \Pr\left[\mathrm{Game}^{\mathsf{AfP\text{-}PRIV}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{U}} = 1\right] - 1\right| \leq \mu(\lambda)$$

5.6.4 More Efficient AfP based on VRF

VRF-based Lottery.

This section introduces a specific lottery mechanism which will be the underlying lottery predicate for the AfP described in the next section. The backbone of the lottery is a VRF scheme VRF as described in [73]. This VRF has the properties of simulatability and unpredictability under malicious key generation which will become useful when arguing about security of the AfP. The VRF scheme is a tuple $(\mathsf{VRF.Gen}, \mathsf{VRF.Prove}, \mathsf{VRF.Verify})$ where $\mathsf{VRF.Gen}(1^\kappa)$ outputs a pair of keys $(\mathsf{VRF.pk}, \mathsf{VRF.sk})$. The $\mathsf{VRF.Prove}$ takes as input a value x and outputs a pair $(y, \pi) \leftarrow \mathsf{VRF.Prove}_{\mathsf{VRF.sk}}(x)$ which is the output value y and the correctness certificate π. The verification is then done by evaluating $\mathsf{VRF.Verify}_{\mathsf{VRF.pk}}(x, y, \pi)$ which outputs 1 iff π attests to the correctness of y as the output of the VRF evaluated on x with key $\mathsf{VRF.sk}$.
We recall the blockchain setup described in Section 5.2.1 where each party P_i is represented by a pair $(\mathsf{Sig.sk}_i, \mathsf{sk}_{L,i})$ associated with public data $(\mathsf{Sig.pk}_i, \mathsf{aux}_i, \mathsf{stake}_i)$. Let aux_i contain a VRF public key $\mathsf{VRF.pk}_i$ as described above and let the lottery secret key be $\mathsf{sk}_{L,i} = (\mathsf{Sig.pk}_i, \mathsf{VRF.sk}_i)$. Finally, we introduce a function $\mathsf{param}(\mathbf{B}, \mathsf{sl})$. This function outputs a tuple $(\{\mathsf{Sig.pk}_i, \mathsf{VRF.pk}_i, \mathsf{stake}_i\}_{i\in[n]}, \eta, \phi)$ associated with the specific blockchain $\mathbf{B}$ and slot sl. Beyond obtaining the public information $(\mathsf{Sig.pk}_i, \mathsf{VRF.pk}_i, \mathsf{stake}_i)$ the function also returns a nonce, η, as well as a public function $\phi(\cdot)$ which on input stake_i computes the threshold for winning the lottery.
The lottery predicate based on the VRF is described in Fig. 5.10.

$(\{\mathsf{Sig.pk}_i, \mathsf{VRF}_{L,i}, \mathsf{stake}_i\}_{i\in[n]}, \eta, \phi) \leftarrow \mathsf{param}(\mathbf{B}, \mathsf{sl})$
$(\mathsf{Sig.pk}_j, \mathsf{VRF.sk}_j) \leftarrow \mathsf{sk}_{L,j}$
$(y, \pi) \leftarrow \mathsf{VRF.Prove}_{\mathsf{sk}_{L,j}}(\mathsf{sl}||\mathsf{R}||\eta)$
if $y < \phi(\mathsf{stake}_j)$ **then**
  if $\mathsf{VRF.Verify}_{\mathsf{vk}_{L,j}}(\mathsf{sl}||\mathsf{R}||\eta, y, \pi) =$ **then**
    return 1
  end if
end if
return 0

Figure 5.10: $\mathsf{lottery}_{\mathsf{VRF}}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,j})$

VRF-based AfP.

With the VRF-based lottery $\mathsf{lottery}_{\mathsf{VRF}}$ in place, we are now ready to introduce the VRF-based AfP. We first note that our general approach of applying a SoK for the knowledge of a secret key still applies. However, using the structure of the lottery, and in particular the VRF, allows for a much more efficient AfP which has applications in most PoS settings as well.
The AfP scheme uses a NIZKPoK which has a setup executed as a part of the blockchain setup such that the CRS is in the genesis block. The algorithms for the scheme are $\pi \leftarrow \mathsf{NIZKPoK.P}(\mathsf{crs}, \mathsf{x}, \mathsf{w})$ and $\{0,1\} \leftarrow \mathsf{NIZKPoK.V}(\mathsf{crs}, \mathsf{x}, \pi)$.

Protocol Π_{AfP} The VRF-based AfP protocol Π_{AfP} is described below.

Authenticate. $\sigma \leftarrow \Pi_{\mathsf{AfP}}.\mathsf{Sign}(\mathbf{B}, \mathsf{sl}, \mathsf{S}, \mathsf{sk}_{L,j}, m)$ To authenticate a message, m, a party first checks that $\mathsf{lottery}_{\mathsf{VRF}}(\mathbf{B}, \mathsf{sl}, \mathsf{S}, \mathsf{sk}_{L,j}) = 1$. It then obtains the output and certificate $(y, \pi_{\mathsf{VRF}}) \leftarrow \mathsf{VRFrf.Prove}_{\mathsf{VRF.sk}_{L,j}}(\mathsf{sl}||\mathsf{R}||\eta)$. Finally, it produces $\pi_{\mathsf{NIZKPoK}} \leftarrow \mathsf{NIZKPoK.P}\{\sigma_{\mathsf{SIG}} \mid \mathsf{Sig.Verify}_{\mathsf{Sig.pk}_j}(\sigma_{\mathsf{SIG}}, m) = 1\}$ which is a NIZK-PoK of a signature produced under $\mathsf{Sig.sk}_j$. It then outputs a tuple $\sigma_{\mathsf{AfP}} \leftarrow (\mathsf{Sig.pk}_j, y, \pi_{\mathsf{VRF}}, \pi_{\mathsf{NIZKPoK}})$

Verify. $\{0,1\} \leftarrow \Pi_{\mathsf{AfP}}.\mathsf{Verify}(\tilde{\mathbf{B}}, \mathsf{sl}, \mathsf{S}, \sigma, m)$ To verify an AfP tag the verifier obtains parameters from the blockchain $(\{\mathsf{Sig.pk}_i, \mathsf{VRF.pk}_i, \mathsf{stake}_i\}_{i\in[n]}, \eta, \phi) \leftarrow \mathsf{param}(\mathbf{B}, \mathsf{sl})$. It then parses the tag as $\sigma_{\mathsf{AfP}} \leftarrow (\mathsf{Sig.pk}_j, y, \pi_{\mathsf{VRF}}, \pi_{\mathsf{NIZKPoK}})$ and gets the VRF verification key $\mathsf{VRF.pk}_j$ for the party that the AfP points to. It then checks the following

1. Makes sure that $\mathsf{VRF.Verify}_{\mathsf{VRF.pk}_j}(\mathsf{sl}||\mathsf{R}||\eta, y, \pi_{\mathsf{VRF}}) = 1$ *i.e.* the VRF output was correctly generated under lottery key of party P_j.
2. Checks that $\mathsf{NIZKPoK.V}(\pi_{\mathsf{NIZKPoK}}, (\mathsf{Sig.pk}_j, m)) = 1$ which verifies the proof of signature knowledge.
3. And $y < \phi(\mathsf{stake}_j)$ which makes sure that the lottery was conducted correctly with the stake of P_j.

If all checks go through, the algorithm outputs 1. Otherwise, it outputs 0.

Theorem 5.6.4. *Let* VRF *be the VRF scheme described in [73] with a secure NIZK-PoK scheme* $\mathsf{NIZKPoK}$. *The protocol* Π_{AfP} *(described above) running in the context of a blockchain protocol* Γ *with underlying lottery* $\mathsf{lottery}_{\mathsf{VRF}}$ *(Fig. 5.10) is an AfP scheme according to Definition 5.6.2.*

Proof. (Sketch) Assume that an adversary $\mathcal{A}$ obtains a non-negligible advantage in $\mathsf{Game}^{\mathsf{EUF\text{-}CMA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{U}}$. In other words, $\mathcal{A}$ is able to forge an AfP tag with noticeable probability. We claim that such an adversary can do at least one of two things:

1. It can forge a signature under $(\mathsf{Sig.sk}_i$, thus violating the EUf-CMA security of the signature scheme.

2. It can produce a convincing proof π_{NIZKPoK} of knowledge of a signature produced with a signature secret key where the corresponding lottery secret key did not win the lottery. Since we assume that only P_i knows the pair $(\mathsf{Sig.sk}_i, \mathsf{sk}_{L,i})$, such a convincing proof must violate the soundness of the $\mathsf{NIZKPoK}$ scheme.

3. It can forge a VRF certificate such that the $\mathsf{VRF.Verify}$ algorithm accepts a certificate π_{VRF} under a different $y' \neq y$ when evaluated with the VRF public key $\mathsf{VRF.pk}$ of the adversary and thus convinces the authenticator. This violates the simulator of the simulatable VRF introduced in [73].

Since we assume that $\mathsf{NIZKPoK}$ is a secure NIZK-PoK scheme and VRF a secure scheme based on the functionality in [73], we conclude that Π_{AfP} is secure with respect to Definition 5.6.2. □

AfP Privacy.

This simple AfP mechanism for a VRF-based lottery predicate does not only satisfy the existential unforgeability definition of an AfP. It has AfP privacy.

Theorem 5.6.5. *The AfP protocol* Π_{AfP} *described above with underlying lottery predicate* $\mathsf{lottery}_{\mathsf{VRF}}$ *running in the context of blockchain* Γ *has AfP privacy.*

Proof (Sketch).
We use the notation $\mathcal{D}_0$ and $\mathcal{D}_1$ for the distribution of outputs when the adversary is put in scenario 0 and scenario 1, respectively. Our aim is to show the existence of a simulator such that $\mathcal{D}_0$ and $\mathcal{D}_1$ are computational indistinguishable.
We introduce 4 hybrids $\{\mathsf{H}_i\}_{i=1,\ldots,4}$ where $\mathsf{H}_1 = \mathcal{D}_0$ and $\mathsf{H}_4 = \mathcal{D}_1$.

H_2 This hybrid is identical to H_1 but we use the simulator of the $\mathsf{NIZKPoK}$ scheme to simulate the proof of signature knowledge that convinces. Due to the security of the $\mathsf{NIZKPoK}$ scheme H_2 and H_1 are indistinguishable to the PPT adversary $\mathcal{A}$.

H_3 The difference from H_2 is that instead of invoking the VRF scheme VRF we are using the simulatability of the construction to output valid proofs.

H_4 This hybrid does not need access to any lottery winning secret keys and thus can be completely simulated by Sim. It is still necessary to observe the distribution of the stake to correctly simulate the output of the oracle $\mathcal{O}_{\mathsf{AfP}}$.

Assume that an adversary can distinguish $\mathcal{D}_0$ and $\mathcal{D}_1$ with non-negligible probability ρ. It implies that there exists an $i \in \{1, 2, 3\}$ such that H_i and H_{i+1} can be distinguished with non-negligible probability at least $\rho/3$. This contradicts the indistinguishability of hybrids. Thus, we conclude that the distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ are computationally indistinguishable due to the simulator Sim obtained through the sequence of hybrids above.

5.6.5 Round and Committee Based YOSO Protocols

Having IND-CCA2 ECW and an EUF-CMA secure and Private AfP, we can establish a round-based YOSO model, where there is a number of rounds $r = 1, 2, \ldots$ and where for each round there are n roles $\mathsf{R}_{r,i}$. We call the role $\mathsf{R}_{r,i}$ "party i in round r". We fix a round length L and associate role $\mathsf{R}_{r,i}$ to slot $\mathsf{sl} = L \cdot r$. This L has to be long enough that in each round the parties executing the roles can decrypt ciphertexts sent to them, execute the steps of the role, compute encryptions to the roles in the next round and post these to the blockchain in time for these to be available to the committee of round $r + 1$ before slot $(r + 1) \cdot L$. Picking such an L depends crucially on the underlying blockchain and network, and we will here simply assume that it can be done for the blockchain at hand.

Using this setup, the roles $\mathsf{R}_{r,i}$ of round r can use ECW and AfP with the aforementioned properties to send secret authenticated messages to the roles $\mathsf{R}_{r+1,i}$ in round $r + 1$. They find their ciphertexts on the blockchain before slot $r \cdot L$, decrypt using ECW, compute their outgoing messages, encrypt using ECW, authenticate using AfP, and post the ciphertexts and AfP tags on the blockchain.

Honest Majority.

In round based YOSO MPC it is critical that we can assume some fraction of honesty in each committee $\mathsf{R}_{r,1}, \ldots, \mathsf{R}_{r,n}$. We discuss here assumptions needed on the lottery for this to hold and how to guarantee it. Assume an adversary that can corrupt parties identified by sk and a lottery assigning parties to roles $\mathsf{R}_{r,i}$. We map the corruption status of parties to roles as follows:

1. If a role $\mathsf{R}_{r,i}$ is won by a corrupted party or by several parties, call the role MALICIOUS. Even if $\mathsf{R}_{r,i}$ is won by two honest parties, they will both execute the role and send outgoing messages, which might violate security.

2. If a role $\mathsf{R}_{r,i}$ is won by exactly one honest party, call it HONEST.

3. If a role $\mathsf{R}_{r,i}$ is not won by any party, call it CRASHED. These roles will not be executed and are therefore equivalent to a crashed party.

Note that because we assume corrupted parties know their lottery witness $\mathsf{sk}_{L,i}$ in our model, we can, in poly-time, extract those witnesses and compute the corruption status of roles. This will be crucial in our reductions. Imagine that a role could be won by an honest party but also by a corrupted party which stays completely silent but decrypts messages sent to the role. If we are not aware of the corrupted party winning the role, we might send a simulated ciphertext to the apparently honest role. The corrupted party also having won the role would be able to detect this. Since any role won by an honest party could also be corrupted by a silent malicious party, simulation would become impossible.

In order to realize YOSO MPC, we will need committees where a majority of the roles are honest according to the description above. We capture this requirement in the definition below.

Definition 5.6.6 (Honest Committee Friendly)**.** *We call a blockchain Γ honest committee friendly if there exist n and H and T such that $H > T$ s.t. we can define a sequence of roles $\mathsf{R}_{r,i}$ for $r = 1, \ldots, \mathsf{poly}(\lambda)$ and $i = 1, \ldots, n$ for a slot sl_r and that for all r it holds that except with negligible probability there are at least H honest roles in $\mathsf{R}_{r,1}, \ldots, \mathsf{R}_{r,n}$ and at most T malicious roles. Furthermore, if an honest party executing $\mathsf{R}_{r,1}, \ldots, \mathsf{R}_{r,n}$ sends a message at sl_r, it is guaranteed to appear on the blockchain before slot sl_{r+1}.*

We are now ready to capture the above discussion using a definition.

Definition 5.6.7 (YOSO Friendly Blockchain)**.** *Let Γ be a blockchain with a lottery predicate $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}_r, \mathsf{R}_{r,i}, \mathsf{sk}_{L,i})$ and let $\mathcal{E} = (\mathsf{Enc}, \mathsf{Dec})$ and $\mathcal{U} = (\mathsf{Sign}, \mathsf{Verify})$ be an EtF and AfP for $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}_r, \mathsf{R}_{r,i}, \mathsf{sk}_{L,i})$, respectively. We call $(\Gamma, \mathcal{E}, \mathcal{U})$ YOSO MPC friendly if the following holds:*

1. *$\mathcal{E}$ is an IND-CCA2 secure EtF (Definition 5.6.1).*
2. *$\mathcal{U}$ is a secure and private AfP (Definitions 5.6.2 and 5.6.3).*
3. *Γ is honest committee friendly (Definition 5.6.6).*

We will later assume a YOSO friendly blockchain, and we argued above that the existence of a YOSO friendly blockchain is a plausible assumption without having given formal proofs of this. It is interesting future work to prove that a concrete blockchain is a YOSO friendly blockchain in a given communication model. We omit this as our focus is on constructing flavours of EtF.

5.7 Construction of EtF from ECW and Threshold-IBE

The key intuition about our construction is as follows: we use IBE to encrypt messages to an arbitrary future $(\mathsf{R}, \mathsf{sl}_{\text{fut}})$ pair. When the winners of the role in slot sl_{fut} are assigned, we let them obtain an ID-specific key for $(\mathsf{R}, \mathsf{sl}_{\text{fut}})$ from the IBE key-generation algorithm using ECW as a channel. Notice that this key-generation happens in the *present* while the encryption could have happened at any earlier time. We generate the key for $(\mathsf{R}, \mathsf{sl}_{\text{fut}})$ in a threshold manner by assuming that, throughout the blockchain execution, a set of committee members each holds a share of the master secret key msk_i.

5.7.1 Construction

We now describe our construction. We assume an encryption to the current winner $\Pi_{\mathsf{ECW}} = (\mathsf{Enc}_{\mathsf{ECW}}, \mathsf{Dec}_{\mathsf{ECW}})$ and a threshold IBE scheme Π_{TIBE}. In the setup stage we assume a dealer acting honestly by which we can assign master secret key shares of the TIBE.

Parameters: We assume that the genesis block B_0 of the underlying blockchain contains all the parameters for Π_{ECW}.

Setup Phase: Parties run the setup stage for the Π_{ECW}. The dealer produces $(\mathsf{mpk}, \mathbf{msk} = (\mathsf{msk}_1, \ldots, \mathsf{msk}_n))$ from TIBE setup with threshold k. Then it chooses n random parties and gives a distinct msk_i to each. All learn mpk.

Blockchain Execution: The blockchain execution we assume is as in Section 5.3. We additionally require that party i holding a master secret key share msk_i broadcasts $\mathsf{ct}^{\mathsf{sk},i}_{(\mathsf{sl},\mathsf{R})} \leftarrow \mathsf{Enc}_{\mathsf{ECW}}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}^i_{(\mathsf{sl},\mathsf{R})})$, whenever the winner of role R in slot sl is defined in the blockchain $\mathbf{B}$, where $\mathsf{sk}^i_{(\mathsf{sl},\mathsf{R})} \leftarrow \Pi_{\mathsf{TIBE}}.\mathsf{IDKeygen}(\mathsf{msk}_i, (\mathsf{sl}, \mathsf{R}))$.

Encryption $\mathsf{Enc}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, m)$**:** Each party generates $\mathsf{ct}_i \leftarrow \Pi_{\mathsf{TIBE}}.\mathsf{Enc}(\mathsf{mpk}, \mathsf{ID} = (\mathsf{sl}, \mathsf{R}), m)$. Output $\mathsf{ct} = (\mathbf{B}, \mathsf{sl}, \mathsf{R}, \{\mathsf{ct}_i\}_{P_i})$.

Decryption $\mathsf{Dec}(\mathbf{B}, \mathsf{ct}, \mathsf{sk})$**:** Party i outputs $\perp$ if it does not have $\mathsf{sk}_{L,i}$ such that $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i}) = 1$ for parameters $\mathbf{B}, \mathsf{sl}, \mathsf{R}$ from ct. Otherwise, it retrieves enough (above threshold) valid ciphertexts $\mathsf{ct}^{\mathsf{sk},j}_{(\mathsf{sl},\mathsf{R})}$ from the current state of the blockchain and decrypts each through Π_{ECW} obtaining $\mathsf{sk}^j_{(\mathsf{sl},\mathsf{R})}$. It then computes $\mathsf{sk}_{(\mathsf{sl},\mathsf{R})} \leftarrow \Pi_{\mathsf{TIBE}}.\mathsf{Combine}(\mathsf{mpk}, (\mathsf{sk}^j_{(\mathsf{sl},\mathsf{R})})_j)$. It finally outputs $m \leftarrow \Pi_{\mathsf{TIBE}}.\mathsf{Dec}(\mathsf{sk}_{(\mathsf{sl},\mathsf{R})}, \mathsf{ct})$.

Resharing. We can ensure that the master secret key is proactively reshared by modifying each party so that msk_i-s are reshared and reconstructed in the evolution of the blockchain.

Correctness. Correctness of the construction follows from the correctness of the underlying IBE and the fact that a winning role will be able to decrypt the id-specific key by the correctness of the ECW scheme.

5.7.2 Security and Proof Intuition

In the following we assume some of the extensions discussed in Section 5.6.

Theorem 5.7.1 (informal)**.** *Let Γ^V be a YOSO MPC friendly blockchain, Π_{TIBE} be a robust secure threshold IBE as in Section 5.2.5 with threshold $n/2$, and Π_{ECW} a secure* IND-CCA2 *ECW. The construction in Section 5.7.1 is a secure EtF.*

At the high level we show security in two steps. We first show the security of our construction for a simplified non-threshold setting with a standard IBE instead of a threshold one with key-sharing. In other words we do not temporarily consider the real case where there is a committee of parties holding a share of the master secret key, but we assume the execution uses a "key provider" oracle holding the master secret key of the IBE scheme. In particular, we define the behavior of oracle $\mathcal{O}^{\text{k-provider}}_{\mathsf{msk}}$ as follows: given in input a blockchain $\mathbf{B}$ and a slot sl (such that the latest slot of $\mathbf{B}$ is sl), it broadcasts a ciphertext for the winner[5] of the slot computed as $\mathsf{ct}^{\mathsf{sk}}_{\mathsf{sl}} \leftarrow \mathsf{Enc}_{\mathsf{ECW}}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{\mathsf{sl}})$ where $\mathsf{sk}_{\mathsf{sl}} \leftarrow \mathsf{IBE.Keygen}(\mathsf{msk}, (\mathsf{sl}, \mathsf{R}))$.

As a second step in the proof we show that, in the threshold-setting (where the master secret key is actually shared), one can obtain an adversary with a comparable advantage in the threshold-setting from an adversary in the non-threshold setting. Intuitively, we can do this because of the low amount of stake the adversary is controlling and the security of threshold-IBE.

[5]This is actually a vector, one for each winner in the slot. For clarity of discussion we just consider the case for one winner. The general case follows straightforwardly.

The non-threshold setting we consider is the same as that in Section 5.7.1 with the following exceptions:

- At the beginning of the run of the blockchain, there is no sharing of the master secret key of the IBE scheme.
- We let the honest parties run exactly as in the other construction, with the exception that they validate and messages related to the shares of the master secret keys, as well as of the secret keys for specific slots.
- We change the way we encapsulate the secret-key for a certain slot. While in Section 5.7.1 we require committee members to each broadcast a ciphertext containing a share of the secret-key for slot sl, here we instead replace that stage with the execution of the following oracle $\mathcal{O}^{\text{k-provider}}_{\mathsf{msk}}$.

  $\underline{\mathcal{O}^{\text{k-provider}}_{\mathsf{msk}}(\mathbf{B}, \mathsf{sl}) :}$

  - $\mathsf{sk}_{\mathsf{sl}} \leftarrow \mathsf{IBE.Keygen}(\mathsf{msk}, (\mathsf{sl}, \mathsf{R}))$
  - $\mathsf{ct}^{\mathsf{sk}}_{\mathsf{sl}} \leftarrow \mathsf{Enc}_{\mathsf{ECW}}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{\mathsf{sl}})$
  - Broadcast $\mathsf{ct}^{\mathsf{sk}}_{\mathsf{sl}}$

Figure 5.11: Hybrid non-threshold setting for proof of security

Finally, our proof considers the case of an adversary with static corruptions, but we point out it can be straightforwardly compiled to a full round and committee YOSO setting as described in Section 5.6.

Proof. We proceed in two steps: first we consider an idealized case where there is no threshold committee; we then show we can prove security of our threshold construction from this setting.

1. The non-threshold case. The simplified setting we will now show security for is in Fig. 5.11. A point on the view of the adversary: we recall that, at any given point in time, a valid blockchain execution contains ciphertexts $\mathsf{ct}^{\mathsf{sk}}_{\mathsf{sl}}$, encrypting slot-specific secret keys for the winner of the slot sl in the chain. In the non-threshold setting, they correspond to the output of the key-provider oracle (in the actual construction, there are more ciphertexts, each containing a share of the key).

Now assume an adversary $\mathcal{A}^{\text{no-thresh}}_{EtF}$ for the EtF security experiment controlling at most an α fraction of the stake with non-negligible success probability in the EtF security experiment. We first to construct an adversary $\mathcal{A}_{\text{IBE}}$ for IBE security using $\mathcal{A}^{\text{no-thresh}}_{EtF}$. Adversary $\mathcal{A}_{\text{IBE}}$ works as follows:

- On receiving the IBE public parameters from the IBE challenger, it injects into blockchain genesis block the IBE's master public. The adversary $\mathcal{A}^{\text{no-thresh}}_{EtF}$ declares a corrupted set of parties S_{corr} and then $\mathcal{A}_{\text{IBE}}$ runs an execution of the blockchain with $\mathcal{A}^{\text{no-thresh}}_{EtF}$ where $\mathcal{A}_{\text{IBE}}$ simulates the honest parties. In this execution $\mathcal{A}_{\text{IBE}}$ acts as key-provider oracle, which it emulates as follows. We distinguish two cases depending on whether the winner of the slot is a corrupted party or an honest one[6]. On query $(\mathbf{B}, \mathsf{sl})$:
  - *if a corrupted party has won* the role for slot sl (i.e. $\mathsf{winners}(\mathbf{B}, \mathsf{sl}, \mathsf{R}) \cap S_{\text{corr}} \neq \varnothing$) then invoke the IBE challenger oracle on identity sl obtaining $\mathsf{sk}_{\mathsf{sl}}$ and broadcast $\mathsf{ct}^{\mathsf{sk}}_{\mathsf{sl}} \leftarrow \mathsf{Enc}_{\mathsf{ECW}}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{\mathsf{sl}})$.

[6]Notice that we can check this for both types of parties as discussed in Section 5.2.1.

– *if a corrupted party has not won* the role for slot sl then broadcast the encryption of a dummy plaintext $\mathsf{ct}_{\mathsf{sl}}^{\mathsf{sk}} \leftarrow \mathsf{Enc}_{\mathsf{ECW}}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathbf{0})$ where $\mathbf{0}$ is a string of zeros of the appropriate length.

The intuitive reason for separating the two cases is that we want to query the same slots that $\mathcal{A}_{EtF}^{\text{no-thresh}}$ wins and no more. In particular we do not want to query the challenge slot sl^* (defined next). Notice, in fact, that only the slots for which the adversary has a corrupted winner will be asked to the IBE key-generation oracle. At the end of this stage, $\mathcal{A}_{EtF}^{\text{no-thresh}}$ will return $(\mathbf{B}, \mathsf{sl}^*, \mathsf{R}, m_0, m_1)$ and $\mathcal{A}_{\text{IBE}}$ will forward $((\mathsf{sl}^*, \mathsf{R}), m_0, m_1)$ to the IBE challenger.

- After receiving a ciphertext ct^* from the IBE challenger, $\mathcal{A}_{\text{IBE}}$ forwards it to $\mathcal{A}_{EtF}^{\text{no-thresh}}$. Then $\mathcal{A}_{\text{IBE}}$ simulates the execution of the blockchain as described above. At the end of the execution $\mathcal{A}_{EtF}^{\text{no-thresh}}$ outputs a guessing bit b^* which $\mathcal{A}_{\text{IBE}}$ forwards to the IBE challenger.

We claim that the advantage of $\mathcal{A}_{\text{IBE}}$ in the IBE experiment is negligibly close to that of $\mathcal{A}_{EtF}^{\text{no-thresh}}$ in the EtF non-threshold experiment (the one without threshold sharing). With that goal in mind, we first show that the inputs we feed to $\mathcal{A}_{EtF}^{\text{no-thresh}}$ and the blockchain execution emulated by $\mathcal{A}_{\text{IBE}}$ is indistinguishable from that in the EtF experiment. Notice that the only difference in the distributions is in the ciphertexts for the non-corrupted winners. If we could distinguish between the two cases, then we could break security of the ECW scheme. Therefore the views of $\mathcal{A}_{EtF}^{\text{no-thresh}}$ in the two cases is indistinguishable. Finally, we lower-bound the success probability of $\mathcal{A}_{\text{IBE}}$. Intuitively, we can observe that two adversaries return the same experiment bit. The only aspect that could impair $\mathcal{A}_{\text{IBE}}$'s success probability compared to $\mathcal{A}_{EtF}^{\text{no-thresh}}$'s is the possibility of having asked the IBE key-generation oracle for the challenge slot sl^*. We observe this does not affect the success probability of $\mathcal{A}_{\text{IBE}}$.

2. Security of threshold construction from non-threshold case. The argument above had a simplified setting where we abstracted out all the threshold aspects of the protocol. This includes the committee holding shares of the master secret key and dealing shares of the slot-specific secret key. We now prove security for the actual threshold scenario building an adversary for our actual (threshold) construction using the adversary for the non-threshold construction (Fig. 5.11).

The threshold adversary $\mathcal{A}_{EtF}^{\text{thresh}}$ needs to emulate the setting for the other adversary where there is a single ciphertexts containing the slot-specific secret key (instead of several containing their shares). It works as follows. First, it corrupts the same parties as $\mathcal{A}_{EtF}^{\text{no-thresh}}$ and executes a blockchain as $\mathcal{A}_{EtF}^{\text{no-thresh}}$ does and broadcasting the same messages it does, with one exception which we now describe. The views of two adversaries (threshold vs non-threshold) differ in only one respect—and so do the two respective blockchains executions. The view of the threshold execution contains ciphertexts of this type for each winning slot sl (we use bracket notation for shares for readability): $\left(\left(\mathsf{ct}_{\mathsf{sl}}^{\text{hon}}[j]\right)_{j \notin S_{\text{corr}}}, \left(\mathsf{ct}_{\mathsf{sl}}^{\text{cor}}[j]\right)_{j \in S_{\text{corr}}} \right)$ These contain the shares for the slot-specific slot sl. The view for the non-threshold execution instead contains a single ciphertext with slot-specific secret key. For a honest slot not corrupted by the adversary, we denote it by $\hat{\mathsf{ct}}_{\mathsf{sl}}^{\text{hon}}$, otherwise we denote it by $\hat{\mathsf{ct}}_{\mathsf{sl}}^{\text{cor}}$. During the blockchain execution $\mathcal{A}_{EtF}^{\text{no-thresh}}$ will expect to see some ciphertext ($\hat{\mathsf{ct}}_{\mathsf{sl}}^{\text{hon}}$/$\hat{\mathsf{ct}}_{\mathsf{sl}}^{\text{cor}}$) whenever a slot is won, which corresponds to a query of $\mathcal{O}_{\mathsf{msk}}^{\text{k-provider}}$. The threshold adversary $\mathcal{A}_{EtF}^{\text{thresh}}$ can emulate this as follows. For every query to $\mathcal{O}_{\mathsf{msk}}^{\text{k-provider}}$:

- *if the slot is won by a honest party*, then broadcast $\hat{\mathsf{ct}}_{\mathsf{sl}}^{\text{hon}} \leftarrow \mathsf{Enc}_{\mathsf{ECW}}(\mathbf{B}, \mathsf{sl}, \mathbf{0})$ for a vector of zeros of the appropriate length.

- *if the slot is won by a corrupted party*, then its view will contain $(\mathsf{ct}_{\mathsf{sl}}^{\text{cor}}[j])_{j\in[n]}$. It can then decrypt them, combine the obtained shares into a slot key $\mathsf{sk}_{\mathsf{sl}}$ and broadcast $\hat{\mathsf{ct}}_{\mathsf{sl}}^{\text{cor}} \leftarrow \mathsf{Enc}_{\mathsf{ECW}}(\mathbf{B}, \mathsf{sl}, \mathsf{sk}_{\mathsf{sl}})$

After receiving challenge messages from $\mathcal{A}_{EtF}^{\text{no-thresh}}$, adversary $\mathcal{A}_{EtF}^{\text{thresh}}$ simply forwards them to its challenger, then continues the execution as above. Finally it outputs the same output guess as $\mathcal{A}_{EtF}^{\text{no-thresh}}$.

We now claim that a successful non-threshold adversary $\mathcal{A}_{EtF}^{\text{no-thresh}}$ for the construction in $Fig.\ 5.11$ would allow $\mathcal{A}_{EtF}^{\text{thresh}}$ to have a similar advantage (up to negligible additive factors). We proceed by a standard hybrid argument. We define the first hybrid H_0 as the output of running the $\mathcal{A}_{EtF}^{\text{thresh}}$ adversary as just described. The "terminal" hybrid H_6 is defined as the output of running the $\mathcal{A}_{EtF}^{\text{no-thresh}}$ adversary. The intermediate hybrids are as follows.

- H_1: like H_0 except that we change one step in how $\mathcal{A}_{EtF}^{\text{thresh}}$ emulates $\mathcal{O}_{\mathsf{msk}}^{\text{k-provider}}$. Specifically, for the case of the honest parties, we now run $(\mathsf{sk}_i^{\mathsf{ID}})_{i\in[n]} \leftarrow \mathsf{Sim}_{\text{kg}}(\mathsf{mpk}, (\mathsf{msk}_i)_{i\in S_{\text{corr}}}, \mathsf{sl})$ to simulate the shares of the honest parties. This simulator exists by key-generation simulation of the threshold IBE scheme. We can then combine all shares to obtain a slot-specific key, encrypt it through ECW and then broadcast the encryption $\hat{\mathsf{ct}}_{\mathsf{sl}}^{\text{hon}}$. We have that $H_0 \approx H_1$ because of the security of ECW, since otherwise we would be able to distinguish encryptions of zeros from encryptions of the (combination of) the simulated slot-specific key shares.

- H_2: as previous item but now, instead of the actual secret shares, we give $\mathcal{A}_{EtF}^{\text{thresh}}$ produced by $\mathsf{Sim}_{\mathsf{msk}}$, the simulator from master secret key shares simulation of the threshold IBE scheme. $H_1 \approx H_2$ follows by the same property.

- H_3: like the previous hybrid, but now we replace the blockchain execution from H_2 with one where we do not use the shares to produce $\hat{\mathsf{ct}}_{\mathsf{sl}}^{\text{hon}}$ and $\hat{\mathsf{ct}}_{\mathsf{sl}}^{\text{cor}}$. Instead we move to a blockchain execution as in Fig. 5.11 with the difference that $\mathcal{O}_{\mathsf{msk}}^{\text{k-provider}}$ has a master secret key computed as follows. Let msk be the master secret key obtained by combining the (simulated) shares msk_i. Then we just run $\mathcal{O}_{\mathsf{msk}}^{\text{k-provider}}$ with this master secret key every time we need to provide a ciphertext for a new winning slot. We have $H_2 \approx H_3$ by definition of $\mathcal{O}_{\mathsf{msk}}^{\text{k-provider}}$, by correctness of the underlying homomorphic secret sharing scheme and the simulation of key-generation evaluations of IBE.

- H_4: as before but we now define msk not as the combination of the shares, but as the output of $\mathsf{Sim}_{\mathsf{msk}}$ on the master public key and the corruption set. $H_3 \approx H_4$ follows by simulation of the master secret-key property of the threshold IBE.

- H_5: Like previous item but now we do not use the key-generation simulator and instead apply the key-generation of the IBE before providing a ciphertext. $H_4 \approx H_5$ again follows by the key-generation simulation of the threshold IBE scheme. Also this is the same as H_6 by construction.

Bounding the Advantage of $\mathcal{A}_{\text{IBE}}$ in Proof of Theorem 5.7.1. Here we formally claim that the advantage of $\mathcal{A}_{\text{IBE}}$ in the IBE experiment is negligibly close to that of $\mathcal{A}_{EtF}^{\text{no-thresh}}$ in the EtF non-threshold experiment:

$$\begin{aligned}\Pr[\mathsf{WinIBE}] &\geq \Pr[\neg\mathsf{QryClgSlot} \wedge \mathsf{WinEtFHyb}] \\ &= (1 - \Pr[\mathsf{QryClgSlot} \mid \mathsf{WinEtFHyb}]) \cdot \Pr[\mathsf{WinEtFHyb}] \\ &\approx (1 - \Pr[S_{\text{corr}} \cap \mathsf{winners}(\mathsf{sl}^*) \neq \varnothing \mid \mathsf{WinEtFHyb}]) \cdot \Pr[\mathsf{WinEtFHyb}] \\ &= \Pr[\mathsf{WinEtFHyb}]\end{aligned}$$

Above the QryClgSlot is the event where $\mathcal{A}_{\text{IBE}}$ queries the challenge slot in the IBE experiment; WinIBE is the event where $\mathcal{A}_{\text{IBE}}$ wins in the IBE experiment; WinEtFHyb is the event where $\mathcal{A}_{EtF}^{\text{no-thresh}}$ wins in the EtF experiment against the non-threshold hybrid model (Fig. 5.11). The first inequality follows by construction of $\mathcal{A}_{\text{IBE}}$. The following ones follow from elementary probability theory and from observing that $\mathcal{A}_{\text{IBE}}$ could query the challenge slot only if that was among the corrupted set (but this does not occur condition on the success of $\mathcal{A}_{EtF}^{\text{no-thresh}}$ by the definition of EtF security).

□

5.8 Blockchain WE versus EtF

In this section we show that an account-based PoS blockchain with sufficiently expressive smart contracts and an EtF scheme for this blockchain implies a notion of witness encryption on blockchains, and *vice versa*. The construction of EtF from BWE is completely straightforward and natural: encrypt to the witness which is the secret key winning the lottery. The construction of BWE from EtF is also straightforward but slightly contrived: it requires that we can restrict the lottery such that only some accounts can win a given role and that the decryptor has access to a constant fraction of the stake on the blockchain and are willing to bind them for the decryption operation. The reason why we still prove the result is that it establishes a connection at the feasibility level. For sufficiently expressive blockchains the techniques allowing to construct EtF and BWE are the same. To get EtF from simpler techniques than those we need for BWE we need to do it in the context of very simple blockchains. In addition, the techniques allowing to get EtF without getting BWE should be such that they prevent the blockchain from having an expressive smart contract layer added. This seems like a very small loophole, so we believe that the result shows that there is essentially no assumptions or techniques which allow to construct EtF which do not also allow to construct BWE. Since BWE superficially looks stronger than EtF the equivalence helps better justify the strong assumptions for constructing EtF.

Definition 5.8.1 (Blockchain Witness Encryption)**.** *Consider PPT algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *in the context of a blockchain* Γ^V *is an BWE-scheme with evolved predicate* evolved *and a lottery predicate* lottery *working as follows:*

Setup. $(\mathsf{pv}, \mathsf{td}) \leftarrow \mathsf{Gen}()$ *generates a public value* pv *and an extraction trapdoor* td*. Initially* pv *is put on* $\mathbf{B}$*.*

Encryption. $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathbf{B}, W, m)$ *takes as input a blockchain* $\mathbf{B}$*, including the public value, a PPT function* W*, the witness recogniser, and a message* m*. It outputs a ciphertext* ct*, a blockchain witness encryption.*

Decryption. $m/\perp \leftarrow \mathsf{Dec}(\tilde{\mathbf{B}}, \mathsf{ct}, \mathsf{w})$ *in input a blockchain state* $\tilde{\mathbf{B}}$*, including the a public value* pv*, a ciphertext* ct *a witness* w*, it outputs a message* m *or* $\perp$*.*

Correctness. An BWE-scheme is correct if for honest parties i and j, PPT function W, and witness w such that $W(\mathsf{w}) = 1$ the following holds with overwhelming probability: if party i runs $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathbf{B}, W, m)$ and party j starts running $\mathsf{Dec}(\tilde{\mathbf{B}}, \mathsf{ct}, \mathsf{w})$ in $\tilde{\mathbf{B}}$ evolved from $\mathbf{B}$, then eventually $\mathsf{Dec}(\tilde{\mathbf{B}}, \mathsf{ct}, \mathsf{w})$ outputs m.

Security. We establish a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. In section 5.2.1 we described how $\mathcal{A}$ and $\mathcal{Z}$ execute a blockchain protocol. In addition, we now let the adversary interact with the challenger in a game $\mathsf{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}}$ which can be summarized as follows.

1. $(\mathsf{pv}, \mathsf{td}) \leftarrow \mathsf{Gen}()$ and put pv on the blockchain.
2. $\mathcal{A}$ executes the blockchain protocol Γ together with $\mathcal{Z}$ and at some round r chooses a blockchain $\mathbf{B}$, a function W and two messages m_0 and m_1 and sends it all to $\mathcal{C}$.
3. $\mathcal{C}$ chooses a random bit b and encrypts the message m_b with the parameters it received and sends ct to $\mathcal{A}$.
4. $\mathcal{A}$ continues to execute the blockchain until some round $\tilde{r}$ where the blockchain $\tilde{\mathbf{B}}$ is obtained and the $\mathcal{A}$ outputs a bit b'.

The adversary wins the game if it succeeds in guessing b with probability notably greater than one half without $W(\mathsf{Extract}(\mathsf{td}, \tilde{\mathbf{B}}, \mathsf{ct}, W)) = 1$.

EtF from BWE.

We first show the trivial direction of getting EtF from BWE. Let $\Pi_{\mathsf{BWE}} = (\mathsf{Gen}_{\mathsf{BWE}}, \mathsf{Enc}_{\mathsf{BWE}}, \mathsf{Dec}_{\mathsf{BWE}})$ be an BWE scheme. Recall that one wins the lottery if $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}) = 1$. We construct a EtF scheme. To encrypt, let W be the function $W(\mathsf{w}) = \mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{w})$ and output $\mathsf{Enc}_{\mathsf{BWE}}(\mathbf{B}, W, m)$. If winning the lottery for $(\mathsf{sl}, \mathsf{R})$ then let w be the secret key winning the lottery and output $\mathsf{Dec}(\tilde{\mathbf{B}}, \mathsf{ct}, \mathsf{w})$. The proof is straightforward.

BWE from EtF.

We now show how to construct BWE from EtF. Let $(\mathsf{Enc}_{\mathsf{EtF}}, \mathsf{Dec}_{\mathsf{EtF}})$ be an EtF scheme. Assume a blockchain with Turing complete smart contracts which can be programmed to send, receive, and reject stake. Assume furthermore that if a constant fraction of the stake is moved to an account then within a polynomial number of slots it will begin winning the lottery with constant probability.

We assume that the contract C of an account is hardcoded into the account when created and cannot be changed. We also need to assume that the blockchain reaches all slot numbers such that there is an independent chance to win at all slot numbers. We also need that only polynomially many slot numbers are reached in polynomial time. We need that the lottery can be filtered such that only certain accounts can win a given role. We need that the filtering can depend on the smart contract put on the account when the account was created.

The construction needs a notion of labelled simulation-sound NIZK proof of knowledge. For such a scheme there is a label connected to a proof and a proof of instance x and label L cannot be mauled into a proof of instance x and label $L' \neq L$. This can generically be constructed from an unlabelled scheme simply by letting the label be part of the instance. Let pv of the BWE scheme be the CRS of the NIZK and let td be the extraction trapdoor of the BWE scheme.

To encrypt proceed as follows.

1. Create a fresh account vk with a smart contract E and with no stake on it. Program E with W hard-coded and such that E is willing to receive calls of the form $(\textsc{Transfer}, \pi, f, F)$ from any other smart contract D. If D has f stake available and π is a proof of knowledge of w such that $W(\mathsf{w}) = 1$ and with label F, then accept a transfer of f stake from D and send them to F.

2. Let filter be the filter which only accepts accounts which have no stake initially and which have smart contracts C of the form that it will only accept stake from the account vk created by the encryptor above.

3. Use $\mathsf{Enc}_{\mathsf{EtF}}$ to encrypt to roles E at slots $2^i + j$ for $i = 1, \ldots, \kappa$ and $j = 1, \ldots, \kappa$. Use the filter filter.

To decrypt create a new account F with a contract accepted by filter. Then use w to transfer stake to F via E. Note that F is allowed to win the lotteries used in the EtF encryptions. No matter when the decryption is performed, the slots of the blockchain will eventually reach the next slot of the form 2^i as at most polynomially many slots were reached already. After this comes κ slots in a row to which the encryptor encrypted using EtF. Each of these is won with a constant probability. Therefore the probability of not decrypting is negligible.

Chapter 6

(Commit-and-Prove) Predictable Arguments with Privacy

In this chapter we present our results on commit-and-prove predictable arguments with privacy, first appeared in [124]. The contents of this chapter are taken almost verbatim from [124].

6.1 Introduction

Interactive proofs (IPs) and arguments introduced by Goldwasser, Micali, and Rackoff [109] are cryptographic protocols that allow a prover to convince a verifier about the veracity of a public statement $x \in \mathcal{L}$, where $\mathcal{L}$ is an NP language. The interaction may consist of several rounds of communication, at the end of which the verifier decides to accept or reject the prover's claim on the membership of x in $\mathcal{L}$. There are two properties required for an IP, namely *completeness* and *soundness*. Completeness means that if $x \in \mathcal{L}$, the honest prover can always convince the honest verifier. Soundness means that for $x \notin \mathcal{L}$ no (even unbounded) malicious prover can convince the honest verifier that $x \in \mathcal{L}$. Argument systems are like IPs, except they are only computationally sound; i.e., it should be *computationally hard* (and not impossible) for a malicious prover to convince the verifier that $x \in \mathcal{L}$. An interactive proof is called *public-coin* if the verifier messages are uniformly and independently random, and *private-coin* otherwise.

Recently, Faonio, Nielsen and Venturi [79] introduced a new property for argument systems called *predictability*. Predictable arguments (PA) are private-coin argument systems where the answer of the prover can be predicted efficiently, given the honest verifier's (private) random coins. The prover in such arguments is deterministic and must be consistent with the unique accepting transcript throughout the entire protocol. Faonio et al. [79] formalized this notion and provided several constructions based on various cryptographic assumptions. They also considered PAs with additional privacy properties, namely a *zero-knowledge* (ZK) property, and showed two transformations from PAs into ZK-PAs, the first in the common reference string (CRS) model, and the second in the non-programmable random oracle (NPRO) model.

6.1.1 Our Contribution

In this paper, we study predictable arguments with privacy properties in more detail. Our results are three-fold:

First, we provide a more efficient construction of ZK-PA in the CRS model. Compared to the generic transformation of [79], the resulting argument is much more efficient although it works only for a restricted class of languages; i.e., all languages that admit SPHFs. This includes all algebraic languages described in Section 6.2.1.

Second, we answer an open problem raised in [79] and show how to construct witness indistinguishable PAs (WI-PA) in the plain model by using non-interactive witness indistinguishable (NIWI) proofs in the plain model. Informally, in order to ensure that the verifier's challenge in the first round is well-formed, we force the verifier to provide a NIWI proof for the statement that "the produced challenge is well-formed". Witness-indistinguishability follows from the soundness of the underlying NIWI and the predictability of the argument. Moreover, we provide a reduction that shows how an adversary breaking the soundness of the WI-PA can be exploited in order to violate the WI property of the underlying NIWI proof system.

Third, motivated by the fact that predictable argument (even without privacy properties) is a strong notion [1], we put forward a relaxation of predictable arguments, namely, commit-and-prove [2] predictable arguments (CPPA) that, except the first message of the prover, all the prover's responses can be predicted. We formalize this notion for the language of dynamic statements of form $\mathsf{x} = (\mathsf{cm}, C, y)$, where cm is the prover's first message, and C is an arbitrary polynomial-size circuit possibly specified by the verifier. In particular, we consider a case where the prover publishes a first message cm, after which the prover can run an unbounded number of predictable arguments for different but correlated statements (cm, C_i, y_i). In contrast to PAs for which efficient construction based on standard assumptions (even without ZK) seems out of reach, we give a construction of ZK-CPPA for any polynomial-size circuit $C \in \mathsf{P}$ in the NPRO model using garbled circuits (GC) and oblivious transfer (OT). Our construction is very similar to the three-round zero-knowledge argument of [92] with the main difference being the reusability of the prover's first message and providing ZK in the non-UC model under milder assumptions.

Applications. To demonstrate the usefulness of (CP)PA with privacy properties, we will give its application in the context of witness encryption. We consider witness encryption schemes with a strong notion of privacy for the decryptor, wherein a malicious encryptor should not learn any information about the decryptor's witness, even after the decryptor reveals the decrypted message. Our motivating applications for this scenario are dark pools and over-the-counter (OTC) markets in which an investor (the encrypting party) is interested to communicate with only those trading parties (potential decryptors) whose financial conditions satisfy some constraint. To realize this application, a recent work by Ngo et al. [138] introduced the notion of *witness key agreement* (WKA) which allows the two sides to agree on a secret key k, given that the trading parties hold a witness that satisfies the desired relation. We show in Section 6.6.1 that the witness encryption (WE) interpretation of our ZK-CPPA construction can be used to realize this application with an efficiency improvement in some aspects.

[1] This follows by the fact that predictable arguments and witness encryption (that only exists based on strong primitives like indistinguishability obfuscation) are equivalent.

[2] We call our notion *commit-and-prove* PA because, roughly speaking, a prover first commits to an input (once and for all) and later proves that an opening for the commitment satisfies some properties of interest. Our name is also inspired by the phrase "commit-and-prove schemes" used in some papers, e.g., [48].

6.1.2 Related work

This paper is a follow-up to the work of Faonio et al. [79] that introduced the notion of predictable arguments of knowledge (PAoK) systems. While PAs are always honest-verifier zero-knowledge, providing zero-knowledge or even the weaker notion of witness-indistinguishability is quite challenging. In [79], the authors show a compiler for constructing ZK-PA in the CRS model and leave the construction of WI-PA in the plain model as an open problem. We answer the open problem and propose more efficient ZK-PAs in the CRS model. A related work is that of Bitansky and Choudhuri [34] who recently constructed deterministic-prover ZK arguments for NP and showed that such arguments imply ZK-PA for NP. Different from [34] who mainly focus on feasibility results and require strong assumptions (e.g., indistinguishability obfuscation) in their construction, our work considers practical solutions in the CRS model. In another related work, Dahari and Lindell [67] studied deterministic-prover honest verifier ZK arguments in the plain model. In the same work, they also constructed full ZK arguments given that the prover has access to a pair of witnesses one of which can be used as a basis for the prover's randomness. This differs from our ZK-PA construction wherein the prover is "truly deterministic" although at the cost of requiring a trusted setup. The recent work of [58] introduced the notion of *Witness Maps*. A Unique Witness Map (UWM) is a cryptographic notion that maps all the witnesses for an NP statement to a single witness in a deterministic way. While UWMs can be seen as deterministic-prover NIWI arguments, they differ from WI-PA in several respects, making the two concepts incomparable. First, WI-PA does not require a trusted setup in the form of a common reference string, whereas UWMs are in the CRS model. Second, we consider WI-PA as an interactive protocol, whereas UWMs are non-interactive. Lastly, although UWMs are deterministic-prover, they are not necessarily predictable.

6.2 Preliminaries

We refer the reader to Section 2.6 for general notation and the definition of bilinear groups.

6.2.1 Algebraic languages.

We refer to algebraic languages as the set of languages associated to a relation that can be described by algebraic equations over abelian groups. To be more precise, let $\mathtt{gpar}$ be some global parameters, generated by a probabilistic polynomial-time algorithm $\mathsf{setup}.\mathtt{gpar}$ which takes the security parameter λ as input. These global parameters can correspond to the description of groups involved in the construction and usually includes the description of a bilinear group. Throughout the paper, we suppose that these global parameters are implicitly given as input to each algorithm.

Let $\mathsf{lpar} = (\mathbf{M}, \boldsymbol{\theta})$ be a set of language parameters generated by a polynomial-time algorithm $\mathsf{setup}.\mathsf{lpar}$ which takes $\mathtt{gpar}$ as input. Here, $\mathbf{M} : \mathbb{G}^\ell \mapsto \mathbb{G}^{n\times k}$ and $\boldsymbol{\theta} : \mathbb{G}^\ell \mapsto \mathbb{G}^n$ are linear maps such that their different coefficients are not necessarily in the same algebraic structures. Namely, in the most common case, given a bilinear group $\mathtt{gpar} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [2]_2)$, they can belong to either $\mathbb{Z}_p$, $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$ as long as the equation $\boldsymbol{\theta}(\mathbf{x}) = \mathbf{M}(\mathbf{x}) \cdot \mathbf{w}$ is "well-consistent".

Formally, for a set $\mathcal{X}_{\mathsf{lpar}}$ that defines the underlying domain, we define an algebraic language $\mathcal{L}_{\mathsf{lpar}} \subset \mathcal{X}_{\mathsf{lpar}}$ as

$$\mathcal{L}_{\mathsf{lpar}} = \left\{ \mathbf{x} \in \mathbb{G} \middle| \exists \mathbf{w} \in \mathbb{Z}_p^k : \boldsymbol{\theta}(\mathbf{x}) = \mathbf{M}(\mathbf{x}) \cdot \mathbf{w} \right\} . \tag{6.1}$$

An algebraic language where $\mathbf{M}$ is independent of $\mathbf{x}$ and $\boldsymbol{\theta}$ is the identity function is called a *linear language*.

Finally, we note that algebraic languages are as expressive as generic NP languages. This is because every binary circuit can be represented by a set of linear equations.

6.2.2 Smooth Projective Hash Function.

Let $\mathcal{L}_{\mathsf{lpar}}$ be a NP language, parametrized by a language parameter lpar, and $\mathcal{R}_{\mathsf{lpar}} \subseteq \mathcal{X}_{\mathsf{lpar}}$ be its corresponding relation. A Smooth projective hash functions (SPHFs, [66]) for $\mathcal{L}_{\mathsf{lpar}}$ is a cryptographic primitive with this property that given lpar and a statement x, one can compute a hash of x in two different ways: either by using a projection key hp and $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathsf{lpar}}$ as $\mathsf{pH} \leftarrow \mathsf{projhash}(\mathsf{lpar}; \mathsf{hp}, \mathsf{x}, \mathsf{w})$, or by using a hashing key hk and $\mathsf{x} \in \mathcal{X}_{\mathsf{lpar}}$ as $\mathsf{H} \leftarrow \mathsf{hash}(\mathsf{lpar}; \mathsf{hk}, \mathsf{x})$. The formal definition of SPHF follows.

Definition 6.2.1. *A SPHF for* $\{\mathcal{L}_{\mathsf{lpar}}\}$ *is a tuple of PPT algorithms* $(\mathsf{setup}, \mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$*, which are defined as follows:*

$\mathsf{setup}(1^\lambda)$**:** *Takes in a security parameter* λ *and generates the global parameters* pp *together with the language parameters* lpar*. We assume that all algorithms have access to* pp*.*

$\mathsf{hashkg}(\mathsf{lpar})$**:** *Takes in a language parameter* lpar *and outputs a hashing key* hk*.*

$\mathsf{projkg}(\mathsf{lpar}; \mathsf{hk}, \mathsf{x})$**:** *Takes in a hashing key* hk*,* lpar*, and a statement* x *and outputs a projection key* hp*, possibly depending on* x*.*

$\mathsf{hash}(\mathsf{lpar}; \mathsf{hk}, \mathsf{x})$**:** *Takes in a hashing key* hk*,* lpar*, and a statement* x *and outputs a hash value* H*.*

$\mathsf{projhash}(\mathsf{lpar}; \mathsf{hp}, \mathsf{x}, \mathsf{w})$**:** *Takes in a projection key* hp*,* lpar*, a statement* x*, and a witness* w *for* $\mathsf{x} \in \mathcal{L}$ *and outputs a hash value* pH*.*

A SPHF needs to satisfy the following properties:

Correctness. It is required that $\mathsf{hash}(\mathsf{lpar}; \mathsf{hk}, \mathsf{x}) = \mathsf{projhash}(\mathsf{lpar}; \mathsf{hp}, \mathsf{x}, \mathsf{w})$ for all $\mathsf{x} \in \mathcal{L}$ and their corresponding witnesses w.

Smoothness. It is required that for any lpar and any $\mathsf{x} \notin \mathcal{L}$, the following distributions are statistically indistinguishable:

$$\left\{(\mathsf{hp}, \mathsf{H}) : \mathsf{hk} \leftarrow \mathsf{hashkg}(\mathsf{lpar}), \mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{lpar}; \mathsf{hk}, \mathsf{x}), \mathsf{H} \leftarrow \mathsf{hash}(\mathsf{lpar}; \mathsf{hk}, \mathsf{x})\right\}$$
$$\left\{(\mathsf{hp}, \mathsf{H}) : \mathsf{hk} \leftarrow \mathsf{hashkg}(\mathsf{lpar}), \mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{lpar}; \mathsf{hk}, \mathsf{x}), \mathsf{H} \leftarrow_{\$} \Omega\right\} .$$

where Ω is the set of hash values.

6.2.3 Predictable Arguments.

Predictable arguments are multi-round interactive protocols where the verifier generates a challenge (which will be sent to the prover) and at the same time it can predict the prover's response to that challenge. Here we recall the formal definition of predictable arguments (PA) [79][3].

Let $\mathcal{RG}$ be a relation generator that takes in a security parameter 1^λ and returns a polynomial-time decidable binary relation $\mathcal{R}_{\mathsf{lpar}}$. For a pair $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathsf{lpar}}$, we call x the statement and w the witness. The set of all possible relations $\mathcal{R}_{\mathsf{lpar}}$ that the relation generator $\mathcal{RG}$ (for a given 1^λ) may output is denoted by $\mathcal{RG}_\lambda$. To make the notation simple, we assume that $\mathcal{R}_{\mathsf{lpar}}$ can be described with a language parameter lpar by which λ can be deduced as well.

Definition 6.2.2 (Predictable Argument (PA))**.** *A predictable argument for a relation $\mathcal{R}_{\mathsf{lpar}}$ (with the corresponding language parameter lpar) is an interactive protocol between a prover P and a verifier V, which can be specified by two algorithms $\Pi_{\mathsf{pa}} = (\mathsf{Chall}, \mathsf{Resp})$ defined as follows:*

*(**Executed by** V): $(c, b) \leftarrow \mathsf{Chall}(\mathsf{lpar}, \mathsf{x})$. The algorithm takes in lpar and a statement x, and returns a challenge c along with a predicted answer b.*

*(**Executed by** P): $a \leftarrow \mathsf{Resp}(\mathsf{lpar}, \mathsf{x}, \mathsf{w}, c)$. The algorithm takes in lpar, a pair of statement-witness (x, w) and a challenge c, and returns an answer a.*

*(**Executed by** V): If $a = b$, V returns acc; otherwise it returns rej.*

We denote by $\langle \mathsf{P}(\mathsf{lpar}, \mathsf{x}, \mathsf{w}), \mathsf{V}(\mathsf{lpar}, \mathsf{x}) \rangle$ an execution between P and V with common inputs $(\mathsf{lpar}, \mathsf{x})$ and prover's secret input w. The success of the prover in convincing the verifier is denoted by $\langle \mathsf{P}(\mathsf{lpar}, \mathsf{x}, \mathsf{w}), \mathsf{V}(\mathsf{lpar}, \mathsf{x}) \rangle = \mathsf{acc}$. Also, we may call (c, b) as both the output of $\mathsf{Chall}()$, or the output of V running $\mathsf{Chall}()$. The same convention holds for a.

We require two properties for a PA: *completeness* and *soundness*.

- **(Perfect) Completeness.** A predictable argument has perfect completeness if for all $\lambda \in \mathbb{N}$, for all $\mathcal{R}_{\mathsf{lpar}} \in \mathcal{RG}_\lambda$, and for all $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathsf{lpar}}$

$$\Pr\left[a = b \ : \ (c, b) \leftarrow \mathsf{Chall}(\mathsf{lpar}, \mathsf{x}); a \leftarrow \mathsf{Resp}(\mathsf{lpar}, \mathsf{x}, \mathsf{w}, c)\right] = 1$$

- ϵ**-Soundness.** For all $\lambda \in \mathbb{N}$, all $\mathsf{x} \notin \mathcal{L}_{\mathsf{lpar}}$, and all PPT adversaries $\mathcal{A}$

$$\Pr\left[a = b \ : \ \mathcal{R}_{\mathsf{lpar}} \leftarrow\!\$\, \mathcal{RG}_\lambda; (c, b) \leftarrow \mathsf{Chall}(\mathsf{lpar}, \mathsf{x}); a \leftarrow \mathcal{A}(\mathsf{lpar}, \mathsf{x}, c)\right] \approx_\lambda \epsilon$$

We call a PA sound if $\epsilon \in \mathsf{negl}(\lambda)$. A PA is secure if it is complete and sound. Furthermore, we say that a PA is *zero-knowledge* (ZK-PA) if there exists a PPT algorithm Sim that computes the predicted answer of any valid statement x without knowing the random coins used in $\mathsf{Chall}()$ nor any witness for x, but only knowing the challenge c. In the case of ZK in the CRS model, the algorithm takes in also a CRS trapdoor τ which is generated by a setup algorithm $(\mathsf{CRS}_\tau, \tau) \leftarrow \mathsf{setup}(1^\lambda)$. For notational simplicity, we assume that in this case lpar contains CRS_τ as well.

[3]We define PAs as one-round protocols. As shown in [79], this is without loss of generality as every ρ-round PA can be squeezed into a one-round PA.

Zero-Knowledge

$(\mathsf{CRS}_\tau, \tau) \leftarrow \mathsf{setup}(1^\lambda); (\mathsf{x}, \mathsf{w}, c) \leftarrow \mathcal{A}(\mathsf{CRS}_\tau, \tau);$

if $\mathcal{R}_{\mathsf{lpar}}(\mathsf{x}, \mathsf{w}) = 0,$ **then return** $0;$

$b \leftarrow\$ \{0,1\}$; **if** $b = 0$ **then** $a \leftarrow \mathsf{Resp}(\mathsf{lpar}, \mathsf{x}, \mathsf{w}, c)$; **else** $a \leftarrow \mathsf{Sim}(\mathsf{lpar}, \mathsf{x}, \tau, c);$

$b' \leftarrow \mathcal{A}(a);$

return $b = b';$

Figure 6.1: Experiment for the definition of Zero-knowledge

- **Zero-Knowledge.** A predictable argument Π is zero-knowledge if there exists a PPT simulator Sim such that for all PPT adversary $\mathcal{A}$, $\Pr[\mathsf{Exp}^{\mathsf{zk}}_{\Pi,\mathsf{Sim}}(\mathcal{A}, \lambda) = 1] \approx_\lambda \frac{1}{2}$, where $\mathsf{Exp}^{\mathsf{zk}}_{\Pi,\mathsf{Sim}}(\mathcal{A}, \lambda)$ is depicted in Fig. 6.1.

In this work, we also consider a weaker version of zero-knowledge, called *witness indistinguishability* (WI) [83] which informally states that the adversarial verifier cannot identify which witnesses are held by the prover.

- **Witness-Indistinguishability.** A predictable argument Π is statistically witness indistinguishable if for any adversary $\mathcal{A}$, for any common statement x, for any witnesses $\mathsf{w}_1, \mathsf{w}_2$ such that $(\mathsf{x}, \mathsf{w}_1) \in \mathcal{R}_{\mathsf{lpar}}, (\mathsf{x}, \mathsf{w}_2) \in \mathcal{R}_{\mathsf{lpar}}$, the following holds:

$$\langle \mathsf{P}(\mathsf{lpar}, \mathsf{x}, \mathsf{w}_1), \mathcal{A}(\mathsf{lpar}, \mathsf{x}) \rangle \approx_\lambda \langle \mathsf{P}(\mathsf{lpar}, \mathsf{x}, \mathsf{w}_2), \mathcal{A}(\mathsf{lpar}, \mathsf{x}) \rangle$$

6.2.4 Oblivious Transfer and Garbling Schemes

We refer the reader to Sections 5.2.3 and 5.2.4 for formal definitions of oblivious transfer protocols and garbling schemes. For our construction in this section, we additionally require a property for oblivious transfers called *sender-extractability* in [92], which informally states that the randomness of the sender is sufficient to reconstruct its input with high probability.

- **Sender-Extractability.** For any security parameter $\lambda \in \mathbb{N}$, for any $b \in \{0,1\}$, for any messages (x^0, x^1), where $x^l \in \{0,1\}^{\mathsf{poly}(\lambda)}$ for $l \in \{0,1\}$, there exist a PPT algorithm Ext such that for $m^R \leftarrow \Pi^R_{\mathsf{OT}}(b; r^R)$, and $m^S \leftarrow \Pi^S_{\mathsf{OT}}(m^R, (x^0, x^1); r^S)$, where $r^R, r^S \in \{0,1\}^{\mathsf{poly}(\lambda)}$, we have

$$\Pr\left[(\bar{x}^0, \bar{x}^1) \neq (x^0, x^1) \ : \ (\bar{x}^0, \bar{x}^1) \leftarrow \mathsf{Ext}(m^R, m^S, r^S)\right] \approx_\lambda 0$$

6.3 TSPHF-based ZK-PAs in the CRS model

As shown in [79], PAs can be constructed from SPHFs, but since the projection key in SPHFs can be generated in a malicious way, they can provide only honest-verifier zero-knowledge property and it is not clear how to construct ZK-PA from standard SPHFs directly. Benhamouda *et al.* [31] defined the notion of *trapdoor SPHFs* (TSPHFs) as an extension of SPHF in which one can verify the correctness of the projection key generation. More in details, a TSPHF comes with

- $\mathsf{Setup}(1^\lambda)$: Run $(\mathsf{CRS}_\tau, \tau) \leftarrow \mathsf{tsetup}(1^\lambda)$ and return $(\mathsf{CRS}_\tau, \tau)$.
- $\mathsf{Chall}(\mathsf{lpar}, \mathsf{x})$:
  - Run $\mathsf{hk} \leftarrow\!\!\$\ \mathsf{hashkg}(\mathsf{lpar})$ and $\mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{lpar}; \mathsf{hk}, \mathsf{x})$.
  - Compute $\mathsf{H} \leftarrow \mathsf{hash}(\mathsf{lpar}; \mathsf{hk}, \mathsf{x})$.
  - Return $(c, b) := (\mathsf{hp}, \mathsf{H})$.
- $\mathsf{Resp}(\mathsf{lpar}, \mathsf{x}, \mathsf{w}, c)$: For $c := \mathsf{hp}$, check if $\mathsf{verHP}(\mathsf{CRS}_\tau, \mathsf{hp}) = 1$, then run $\mathsf{pH} \leftarrow \mathsf{projhash}(\mathsf{lpar}; \mathsf{hp}, \mathsf{x}, \mathsf{w})$ and return $a := \mathsf{pH}$.
- $\mathsf{Sim}(\mathsf{lpar}, \mathsf{x}, \tau, c)$: Parse $c := \mathsf{hp}$ and return $\mathsf{tH} \leftarrow \mathsf{thash}(\mathsf{lpar}; \mathsf{hp}, \mathsf{x}, \tau)$.

Figure 6.2: ZK-PA Π_{zkpa} from TSPHFs.

three additional algorithms $(\mathsf{tsetup}, \mathsf{verHP}, \mathsf{thash})$. tsetup outputs a CRS CRS_τ with a trapdoor τ. The trapdoor τ can be used by thash to compute the hash value of any statement x (only by knowing public hp). The algorithm verHP takes in a key hp and the CRS CRS_τ, and outputs 1 if hp is a valid projection key. The properties a TSPHF must verify are the same as SPHF, except the smoothness property is no longer statistical but computational as hp should now contain enough information to compute the hash of any statement. Moreover, a TSPHF should satisfy zero-knowledge property which informally states that for any statement x with valid witness w, the projected hash value $\mathsf{pH} \leftarrow \mathsf{projhash}(\mathsf{lpar}; \mathsf{hp}, \mathsf{x}, \mathsf{w})$ should be indistinguishable from the trapdoor hash value $\mathsf{tH} \leftarrow \mathsf{thash}(\mathsf{lpar}; \mathsf{hp}, \mathsf{x}, \tau)$. For a more formal definition of TSPHFs, We refer the reader to [31].

In this section, we show the connection between ZK-PAs and TSPHFs [30], namely we construct ZK-PA for a relation $\mathcal{R}_{\mathsf{lpar}}$ given a TSPHF for the same relation. Different from [79], the relation $\mathcal{R}_{\mathsf{lpar}}$ here is identical. This is because [79] considers the connection for the knowledge-sound PAs (and extractable SPHFs) whereas here we only consider soundness and (computational) smoothness. As a direct result of this, we obtain ZK-PA for all languages that admit TSPHFs (i.e., *algebraic languages*).

Construction of ZK-PA from TSPHFs.

We are now ready to present our construction of ZK-PAs from TSPHFs. Let $\Pi_{\mathsf{tsphf}} = (\mathsf{setup}, \mathsf{tsetup}, \mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash}, \mathsf{verHP}, \mathsf{thash})$ be a TSPHF for $\mathcal{L}_{\mathsf{lpar}}$. The construction of $\Pi_{\mathsf{zkpa}} = (\mathsf{Setup}, \mathsf{Chall}, \mathsf{Resp}, \mathsf{Sim})$ in the CRS model is given in Fig. 6.2.

Theorem 6.3.1. *If the TSPHF Π_{tsphf} is correct, (computationally) smooth and zero-knowledge, then Π_{zkpa} in Fig. 6.2 is secure and zero-knowledge.*

Proof. The correctness of Π_{zkpa} is trivial and follows directly from the correctness of the TSPHF. Here we only give a sketch of the proofs for soundness and ZK.
(Soundness). To show soundness, let $\mathcal{A}$ be a PPT adversary that breaks the soundness, i.e., $\mathcal{A}$ outputs the predicted answer for a chosen $\mathsf{x} \notin \mathcal{L}_{\mathsf{lpar}}$ with non-negligible probability. We construct

a PPT algorithm $\mathcal{B}$ that breaks the smoothness of the underlying TSPHF. Given $(\mathsf{hp}, \mathsf{H})$ as input, $\mathcal{B}$ proceeds as follows: it first runs $\mathcal{A}$ on the security parameters to receive x. Next, it runs $\mathcal{A}$ on input $(\mathsf{x}, \mathsf{hp})$ and receives the answer a. Now depending on whether $a = \mathsf{H}$ or not, $\mathcal{B}$ can decide if H is a hash value or random. This breaks the smoothness property.
(ZK). The ZK property can be shown by a straightforward reduction to the ZK property of the underlying TSPHF. To do so, let $\mathcal{A}$ be an efficient adversary against the ZK property of Π_{zkpa}. We construct an efficient algorithm $\mathcal{B}$ against the ZK property of Π_{tsphf} as follows: $\mathcal{B}$ runs $(\mathsf{CRS}_\tau, \tau) \leftarrow \mathsf{setup}(1^\lambda)$ and sends $(\mathsf{CRS}_\tau, \tau)$ to $\mathcal{A}$. Upon receiving $(\mathsf{x}, \mathsf{w}, c)$ from $\mathcal{A}$, $\mathcal{B}$ sends the same tuple $(\mathsf{x}, \mathsf{w}, c)$ to the challenger. Given the challenge a, $\mathcal{B}$ finally runs $\mathcal{A}(a)$ and return the same guess as $\mathcal{A}$. It is easy to see that $\mathcal{B}$ can now distinguish a real answer from a simulated one with the same advantage as $\mathcal{A}$'s advantage. This completes the proof. □

Instantiation and Efficiency Evaluation. Given the above connection, one can now obtain a secure ZK-PA for any algebraic language $\mathcal{L}_{\mathsf{lpar}}$ with $\mathsf{lpar} = (\mathbf{M}, \boldsymbol{\theta})$ (see Eq. (6.1)) in the bilinear setting based on the efficient construction of TSPHF in [30]. For the sake of completeness, we provide the construction in Fig. 6.3. The resulting ZK-PA is sound under the DDH assumption in $\mathbb{G}_2$ (See [30], Appendix E.3 for the security proof). To evaluate efficiency, we note that compared to the original construction of ZK-PA in [79], the above construction is more efficient as it only has one more group element in the challenge c (compared to the non-zk construction of PA), whereas the idea of adding a NIZK proof for the well-formedness of c in [79] has at least a linear overhead in the size of c [4].

- $\mathsf{Setup}(1^\lambda)$: $\tau \leftarrow\!\$\ \mathbb{Z}_p$; **return** $(\mathsf{CRS}_\tau = [\tau]_2, \tau)$.
- $\mathsf{Chall}(\mathsf{lpar}, \mathbf{x})$: $\boldsymbol{\alpha} \leftarrow\!\$\ \mathbb{Z}_p^n$; $[\gamma]_1 \leftarrow \boldsymbol{\alpha}^\top [\mathbf{M}(\mathbf{x})]_1$; $[\xi]_2 \leftarrow \boldsymbol{\alpha} [\tau]_2$; $c \leftarrow ([\gamma]_1, [\xi]_2)$; $b \leftarrow \hat{e}(\boldsymbol{\alpha}^\top [\boldsymbol{\theta}(\mathbf{x})]_1, [1]_2)$; **return** (c, b).
- $\mathsf{Resp}(\mathsf{lpar}, \mathbf{x}, \mathbf{w}, c)$: parse c as $([\gamma]_1, [\xi]_2)$; **if** $([\xi]_2 \notin \mathbb{G}_2^n \vee \hat{e}([\gamma]_1, \mathsf{CRS}_\tau) \neq \hat{e}([\mathbf{M}]_1, [\xi]_2))$ **return** $\perp$; **else return** $a \leftarrow \hat{e}([\gamma]_1 \mathbf{w}, [1]_2)$.
- $\mathsf{Sim}(\mathsf{lpar}, \mathbf{x}, \tau, c)$: parse c as $([\gamma]_1, [\xi]_2)$; **return** $\tau^{-1} \hat{e}([\boldsymbol{\theta}(\mathbf{x})]_1, [\xi]_2)$.

Figure 6.3: Construction of ZK-PA from [30].

Remark 6.3.2 (Non-Blackbox Construction in the plain model). Recently, Abdolmaleki et al. [6] show how one can use non-blackbox techniques to construct a subversion-resistant variant of smooth projective hash functions. Following a similar approach directly yields the construction of ZK-PA in the plain model, thus giving another way to circumvent the [105] impossibility using non-blackbox techniques. The key idea is to rely on the existence of an efficient (non-blackbox) extractor that—after checking the well-formedness of c— can extract a function of the verifier's randomness (e.g., $[\boldsymbol{\alpha}]_2$) by which one can efficiently compute the predictable answer b.

Remark 6.3.3. In their recent work, Bitansky and Choudhuri [34] also construct ZK-PA for all NP. Their construction, however, mainly focuses on a *feasibility* result rather than efficiency,

[4] Here we are assuming that the security of the construction should remain under standard and falsifiable assumptions as it is easy to construct succinct NIZKs based on non-falsifiable assumptions.

and requires strong assumptions such as indistinguishability obfuscation. Moreover, while the zero-knowledge simulator in their construction is non-black-box which is inherent in the plain model, we rather focus on more efficient constructions in the CRS model.

6.4 Witness-Indistinguishable Predictable Arguments

Due to a classical impossibility result [105], a prerequisite for constructing 2-message ZK proof systems based on black-box techniques is a common reference string (CRS)—a string generated by a trusted party to which both prover and verifier have access. Requiring such a trust model may however be overkill for some applications where a weaker notion of privacy such as witness indistinguishability (WI) is sufficient. Weaker than ZK property, this property states that for any two possible witnesses $\mathsf{w}_1, \mathsf{w}_2$, an adversary cannot distinguish proofs generated by w_1 from the proofs generated by w_2. Given a PA $\Pi_{\mathsf{pa}} = (\mathsf{Chall}, \mathsf{Resp})$ for an NP language $\mathcal{L}$, we show how to construct a WI-PA $\Pi_{\mathsf{wipa}} = (\mathsf{Chall}', \mathsf{Resp}')$ for the same language. At first it may seem that regardless of which witness is used by the prover when running Resp, it has the same functionality since all the witnesses return the same (predicted) answer. This argument is however not true: while for an honestly-generate challenge, Resp behaves the same regardless of which valid witness is used, this might not be true for maliciously generated challenges. To circumvent this issue, the key idea is to require the verifier to prove that the challenge is indeed generated from a proper run of Chall with some randomness. This should be done without breaking the soundness, meaning the secret coins of the verifier should be kept hidden from the prover. To this end, we will use a NIWI proof system as an ingredient, through which the verifier proves the following statement: there exists a random string α, such that $c = \mathsf{Chall}(\mathsf{lpar}, \mathsf{x}; \alpha)$. The prover first checks if the NIWI proof verifies and if so, computes the predicted answer as before.

Since we use a NIWI proof system in the plain model as an ingredient of our construction, below we recall the definition of WI for such proof systems. We note that a construction of NIWI in the plain model for all NP languages and based on standard assumptions is presented in [118].

Definition 6.4.1. *Let $\Pi_{\mathsf{niwi}} = (\mathsf{P}_{\mathsf{niwi}}, \mathsf{V}_{\mathsf{niwi}})$ be a non-interactive proof system for a language $\mathcal{L}_{\mathsf{lpar}}$. We say that Π_{niwi} is computationally witness-indistinguishable if for all $(\mathsf{x}, \mathsf{w}_1, \mathsf{w}_2)$ such that $(\mathsf{x}, \mathsf{w}_1) \in \mathcal{R}_{\mathsf{lpar}}$ and $(\mathsf{x}, \mathsf{w}_2) \in \mathcal{R}_{\mathsf{lpar}}$, and for all PPT adversaries $\mathcal{A}$,*

$$\Pr\left[\mathcal{A}(\pi) = 1 \ : \ \pi \leftarrow \mathsf{P}_{\mathsf{niwi}}(\mathsf{lpar}, \mathsf{x}, \mathsf{w}_1)\right] \approx_\lambda \Pr\left[\mathcal{A}(\pi) = 1 \ : \ \pi \leftarrow \mathsf{P}_{\mathsf{niwi}}(\mathsf{lpar}, \mathsf{x}, \mathsf{w}_2)\right]$$

6.4.1 Our Construction

Let $\Pi_{\mathsf{pa}} = (\mathsf{Chall}, \mathsf{Resp})$ be a predictable argument for language $\mathcal{L}_{\mathsf{lpar}}$, and Π_{niwi} be a non-interactive computational WI proof system in the plain model for the language of statements c for which there exists α such that $c = \mathsf{Chall}(\mathsf{lpar}, \mathsf{x}; \alpha)$. We construct a WI-PA $\Pi_{\mathsf{wipa}} = (\mathsf{Chall}', \mathsf{Resp}')$ for $\mathcal{L}_{\mathsf{lpar}}$ as depicted in Fig. 6.4. The completeness of the construction follows straightforwardly from the completeness of Π_{pa}. We prove soundness and WI in the next theorem.

Theorem 6.4.2. *The construction in Fig. 6.4 is a statistical witness-indistinguishable predictable argument in the plain model.*

- $\mathsf{Chall}'(\mathsf{lpar}, \mathsf{x})$: the verifier computes $(c, b) \leftarrow \mathsf{Chall}(\mathsf{lpar}, \mathsf{x}; \alpha)$ and sends the challenge c along with a NIWI proof π for the existence of α such that c is the first output of $\mathsf{Chall}(\mathsf{lpar}, \mathsf{x}; \alpha)$.
- $\mathsf{Resp}'(\mathsf{lpar}, \mathsf{x}, \mathsf{w}, c, \pi)$: the prover first checks the NIWI proof π. If π verifies, the prover computes $a \leftarrow \mathsf{Resp}(\mathsf{lpar}, \mathsf{x}, \mathsf{w}, c)$ and returns a.

Figure 6.4: Construction of WI-PA

Proof. **Soundness.** Let $\mathsf{x} \notin \mathcal{L}_{\mathsf{lpar}}$ and $\mathcal{A}$ be an efficient adversary that breaks soundness of Π_{wipa} by convincing the honest verifier V with non-negligible probability ε. I.e., $\varepsilon(n) \geq \frac{1}{p(n)}$ for some polynomial p and for infinitely many n's. Denoting this set by N, we restrict ourselves to $n \in N$ from now on. This indicates that there exists a first message c from V on which $\mathcal{A}$ convinces V with
Define a set S as follows:

$$S = \left\{ b : \Pr\left[\mathcal{A}(\mathsf{lpar}, \mathsf{x}, c) = b | (c, b) \leftarrow \mathsf{Chall}(\mathsf{lpar}, \mathsf{x})\right] \geq \frac{\varepsilon}{2} \right\}$$

Fix some $b_0 \in S$ and define $S_0 \subseteq S$ as

$$S_0 = \left\{ b \in S : \Pr\left[\mathcal{A}(\mathsf{lpar}, \mathsf{x}, c) = b | (c, b_0) \leftarrow \mathsf{Chall}(\mathsf{lpar}, \mathsf{x})\right] \geq \frac{\varepsilon}{4} \right\}.$$

Since $b_0 \in S$, we have that $\Pr\left[\mathcal{A}(\mathsf{lpar}, \mathsf{x}, c) = b_0 | (c, b_0) \leftarrow \mathsf{Chall}(\mathsf{lpar}, \mathsf{x})\right] \geq \frac{\varepsilon}{2}$ and therefore $|S_0| \cdot \frac{\varepsilon}{4} \leq 1 - \frac{\varepsilon}{2}$, which consequently implies that $|S_0| \leq \frac{4}{\varepsilon}$. Now, the fact that ε is non-negligible indicates that S_0 is bounded by a polynomial. On the other hand, we have that $\Pr[b \in S] \geq \frac{\varepsilon}{2}$, and that S is exponential in the security parameter λ. This means that there should exist $b_1 \in S$ such that $b_1 \notin S_0$. We now construct a non-uniform PPT adversary $\mathcal{B}$ that breaks the witness-indistinguishability of Π_{niwi}. Let $aux = (\alpha_0, \alpha_1, b_0, b_1)$ be such that $(c, b_0) \leftarrow \mathsf{Chall}(\mathsf{lpar}, \mathsf{x}; \alpha_0)$ and $(c, b_1) \leftarrow \mathsf{Chall}(\mathsf{lpar}, \mathsf{x}; \alpha_1)$. Given aux as advice, $\mathcal{B}$ proceeds as follows: it first returns $(c, (b_0, \alpha_0), (b_1, \alpha_1))$ to the WI challenger and obtains a proof π. Next, $\mathcal{B}$ calls $\mathcal{A}$ on input (π, c) and returns i when it receives b_i from $\mathcal{A}$. Note that for π that is computed using (r_0, b_0), $\mathcal{A}$ returns b_1 with probability at most $\frac{\varepsilon}{4}$, whereas for π computed by (r_1, b_1), $\mathcal{A}$ returns b_1 with probability at least $\frac{\varepsilon}{2}$. This makes $\mathcal{B}$ a successful adversary in breaking WI.
WI. Let V^* be an adversary against WI property of Π_{wipa} and $(\mathsf{x}, \mathsf{w}_1, \mathsf{w}_2)$ be such that $(\mathsf{x}, \mathsf{w}_1), (\mathsf{x}, \mathsf{w}_2) \in \mathcal{R}_{\mathsf{lpar}}$. It follows from (statistical) soundness of the NIWI proof that V^*'s first message is computed correctly with overwhelming probability. This together with predictability of the argument indicates that the answer from the prover is unique regardless of which witness is used and thus completes the proof. □

6.5 Commit-and-Prove Predictable Arguments

We study a relaxed notion of predictability in interactive argument systems which consists of two phases: In phase 1 (commitment phase), the prover commits to its witness once for all and sends the commitment to the verifier. In phase 2 (challenge-response phase), the prover and the

verifier engage in a predictable argument protocol, where the verifier's challenges may depend on the commitment in such a way that the prover's responses can be predicted by the verifier. The type of relations we consider are of the following form: a statement $\mathsf{x} = (\mathsf{cm}, C, y)$ and a witness (w, d) are in the relation (i.e., $(\mathsf{x}, (\mathsf{w}, \mathsf{d})) \in \mathcal{R}$) iff "$\mathsf{cm}$ commits to w by randomness d, and $C(\mathsf{w}) = y$". Here C is a circuit in some polynomial-size circuit class $\mathcal{C}$ and y is the expected output of the circuit.

Definition 6.5.1 (Commit-and-Prove Predictable Arguments)**.** *Let $\mathcal{C}$ be a class of polynomial-sized circuits. A commit-and-prove predictable argument for $\mathcal{C}$ is a multi-round protocol (between a prover P and a verifier V) which consists of three algorithms $\Pi_{\mathsf{cppa}} = (\mathsf{Commit}, \mathsf{Chall}, \mathsf{Resp})$:*

Commitment phase *(**executed by** P):* $\mathsf{cm} \leftarrow \mathsf{Commit}(\mathsf{w}; \mathsf{d})$ *on input a value* w*, generates a commitment* cm *by using some randomness* d*.*

Interaction phase. *Each round proceeds as follows:*

- *(**Executed by** V):* $(c, b) \leftarrow \mathsf{Chall}(\mathsf{cm}, C, y)$ *on input a statement* (cm, C, y) *such that $C \in \mathcal{C}$, generates a challenge c and a predicted answer b.*
- *(**Executed by** P):* $a \leftarrow \mathsf{Resp}(\mathsf{cm}, C, \mathsf{w}, \mathsf{d}, c)$ *on input a commitment* cm*, a circuit $C \in \mathcal{C}$, the committed value* w*, the randomness* d*, returns a response a.*

V *accepts the proof iff $a = b$ in all rounds.*

We call a CPPA as a ρ-round CPPA if the interaction phase consists of ρ rounds. A CPPA should satisfy *completeness* and *soundness* as defined below:

(Perfect) Completeness. An honest prover with a statement $\mathsf{x} = (\mathsf{cm}, C, y)$ and witness (w, d) such that (w, d) opens the commitment (i.e., $\mathsf{cm} = \mathsf{Commit}(\mathsf{w}; \mathsf{d})$), and $C(\mathsf{w}) = y$ can always convince the verifier with overwhelming probability. More precisely, a CPPA has perfect completeness if for all $\lambda \in \mathbb{N}$, for all $C \in \mathcal{C}$, and for all $(\mathsf{x} = (\mathsf{cm}, C, y), (\mathsf{w}, \mathsf{d})) \in \mathcal{R}$

$$\Pr\left[a = b \ : \ (c, b) \leftarrow \mathsf{Chall}(\mathsf{cm}, C, y); a \leftarrow \mathsf{Resp}(\mathsf{cm}, C, \mathsf{w}, \mathsf{d}, c)\right] = 1$$

ϵ-Soundness. For all $\lambda \in \mathbb{N}$, and all (stateful) PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$

$$\Pr\left[\begin{matrix} a = b \wedge \\ C(\mathsf{w}) \neq y \end{matrix} \ : \ \begin{matrix} (\mathsf{w}, \mathsf{d}, C, y) \leftarrow \mathcal{A}_1(1^\lambda); \mathsf{cm} \leftarrow \mathsf{Commit}(\mathsf{w}; \mathsf{d}) \\ (c, b) \leftarrow \mathsf{Chall}(\mathsf{cm}, C, y); a \leftarrow \mathcal{A}_2(\mathsf{w}, \mathsf{d}, C, y, c) \end{matrix}\right] \approx_\lambda \epsilon$$

We call a CPPA sound if $\epsilon \in \mathsf{negl}(\lambda)$. A CPPA is secure if it is correct and sound. Similar to PAs, one can show that CPPAs can also be made extremely laconic in terms of both round complexity and proof complexity. Specifically, the same technique in [79] can be used to collapse any ρ-round CPPA into a single round CPPA.

In this work, we only focus on CPPA protocols with the zero-knowledge property. A CPPA is *zero-knowledge* (ZK-CPPA) if there exists a PPT algorithm Sim that computes the predicted answer of any valid statement x without knowing the random coins used by $\mathsf{Chall}()$ nor any witness for x, but only knowing the challenge c. Since our construction of ZK-CPPA is in the non-programmable random oracle (NPRO) model, we define this property in this model.

Definition 6.5.2 (Zero-knowledge CPPA in the NPRO model)**.** *We say that a CPPA* $(\mathsf{Commit}, \mathsf{Chall}, \mathsf{Resp})$ *for a class of circuits* $\mathcal{C}$ *satisfies the* zero-knowledge *property in the NPRO model if for any PPT adversary* $\mathcal{A}$*, there exists a PPT simulator* Sim *such that for all PPT distinguisher* $\mathcal{D}$*, for all* $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$*, and all auxiliary inputs* $z \in \{0,1\}^*$*, we have:*

$$\max_{\mathcal{D},z} \Big| \Pr[\mathcal{D}^H(\mathsf{x}, \tau, z) = 1 : \tau \leftarrow (\mathsf{P}^H(\mathsf{x}, \mathsf{w}) \leftrightarrows \mathcal{A}^H(\mathsf{x}, z))] - \Pr[\mathcal{D}^H(\mathsf{x}, \tau, z) = 1 : \tau \leftarrow \mathsf{Sim}^H(\mathsf{x}, z)] \Big| \leq \mathsf{negl}(|\mathsf{x}|)$$

Where P *and* $\mathcal{A}$ *are respectively the prover and the (malicious) verifier running the CPPA protocol, and* $\mathsf{P}^H(\mathsf{x}, \mathsf{w}) \leftrightarrows \mathcal{A}^H(\mathsf{x}, z)$ *denotes the random variable corresponding to a protocol transcript on input* (x, w)*.*

We now give our construction of ZK-CPPA for all polynomial-size circuits P in the NPRO model. The construction is similar to the three-round ZK protocol of [92], with the difference that the first message in our protocol is reusable. Moreover, here we only focus on providing ZK property as defined above, whereas the construction of [92] shows ZK in the UC model.

6.5.1 ZK-CPPA based on garbled circuits and oblivious transfer

Let $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode}, \mathsf{Verify})$ be a garbled circuit with correctness, authenticity, and verifiability, and $\Pi_{\mathsf{OT}} = (\Pi^R_{\mathsf{OT}}, \Pi^S_{\mathsf{OT}}, \Pi^O_{\mathsf{OT}})$ be a sender-extractable oblivious transfer protocol that realizes $\mathcal{F}_{\mathsf{OT}}$. At a high level, the construction proceeds as follows. The prover P with witness $\mathsf{w} = (\mathsf{w}_1, \ldots, \mathsf{w}_n) \in \{0,1\}^n$ plays the role of the receiver in n instances of the OT protocol and commits to its witness bits by providing w_j as input to the j-th instance of Π_{OT}. Let $m^R_j \leftarrow \Pi^R_{\mathsf{OT}}(\mathsf{w}_j; r^R_j)$ and define cm and d as the set of $\{m^R_j\}_{j\in[n]}$ and $\{r^R_j\}_{j\in[n]}$, respectively. For a circuit-value pair (C, y) of the verifier's choice, let $\hat{C}$ be a circuit that realizes the following relation $\mathcal{R}$: $\mathcal{R}(\mathsf{x} = (\mathsf{cm}, C, y), (\mathsf{w}, \mathsf{d})) = 1$ iff (w, d) open cm and $C(\mathsf{w}) = y$. The verifier V constructs a GC $\mathbf{C}$ for $\hat{C}$ and sends it along with the second message of the OT as the challenge c. Moreover, V sets the predicted answer b to be the output 1-key k^1 of the final gate in the circuit. Now, P with a valid witness (w, d) evaluates $\mathbf{C}$ and sends the obtained garbled output $a = k^1$ as the predicted answer. It is not hard to see that this construction results in a CPPA. To additionally ensure ZK property, we follow the same approach as [92] by enforcing V to also provide a ciphertext $ct = H(\mathsf{k}^1) \oplus r$, where H is a random oracle and r is the randomness used by V to produce the second message of the OT. When P computes k^1, she first recovers r and then computes all the labels by executing the extractor Ext guaranteed by the sender-extractability property. Finally, P verifies if the garbled circuit has been constructed correctly and if so, she sends the predicted answer $a = \mathsf{k}^1$ to V. The resulting protocol Π_{cppa} is described in Fig. 6.5. The proof idea is similar in spirit to the proof of Theorem 4.2 in [92]. We give a proof sketch here.

Theorem 6.5.3. *Let* GC *be a correct, authentic, and verifiable garbling scheme,* Π_{OT} *be a sender-extractable OT protocol that securely implements* $\mathcal{F}_{\mathsf{OT}}$*, and* H *be a random oracle. The protocol* Π_{cppa} *in Fig. 6.5 is a secure and zero-knowledge commit-and-prove predictable argument as defined in Definitions 6.5.1 and 6.5.2.*

Sketch. Completeness follows straightforwardly by the correctness property of the underlying OT and the garbling scheme.

In order to show soundness, let us consider a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and assume that $(\mathsf{w}, \mathsf{d}, C, y)$ is a tuple returned by $\mathcal{A}_1$ that corresponds to a false statement. That is, $\mathsf{x} = (\mathsf{cm}, C, y)$, where $\mathsf{cm} = \mathsf{Commit}(\mathsf{w}; \mathsf{d})$ and $C(\mathsf{w}) \neq y$. We show that for $(c, b) \leftarrow \mathsf{Chall}(\mathsf{cm}, C, y)$, if $\mathcal{A}_2$ having c can compute the predicted answer b, then one can either break the sender security of the underlying OT protocol, or the authenticity of the garbling scheme. To show this reduction, we first note that b is the correct label k^1. Now, given that $C(\mathsf{w}) \neq y$, there can be two cases where $\mathcal{A}_2$ can output k^1 with non-negligible probability. In the first case, $\mathcal{A}_2$ outputs k^1 by the ability of computing invalid labels $k_j^{1-\mathsf{w}_j}$ that does not correspond to its committed value. It is not hard to see that such $\mathcal{A}_2$ can be used to break OT sender security. The reduction $\mathcal{B}$ proceeds as follows: $\mathcal{B}$ first computes a garbled circuit $\mathbf{C}$ and sends the labels to the OT challenger. Next, it extracts $\mathcal{A}_2$'s input w and forwards it as the choice bits of the receiver. The OT challenger computes the sender's message either by invoking a real sender, or by invoking the simulator, and sends it to the reduction who further forwards to $\mathcal{A}_2$ together with $\mathbf{C}$ and a random T. Now, since $\mathcal{A}_2$ can compute k^1 only in the real execution of Π_{OT}, a successful $\mathcal{A}_2$ with non-negligible probability ϵ implies that $\mathcal{B}$ can distinguish the real and simulated view of the OT protocol with probability at least ϵ. In the second case, where $\mathcal{A}_2$ does not use invalid labels but computes the correct k^1, it is straightforward to construct an adversary $\mathcal{B}$ that breaks the authenticity of the underlying garbling scheme by forging k^1 for a given garbled circuit $\mathbf{C}$.

We now argue that Π_{cppa} is zero-knowledge in the NPRO model. Let V^* be a PPT adversary against the ZK property. We construct an efficient simulator Sim that simulates the protocol as follows. Sim observes V^*'s calls to the random oracle, so that for every query $H(u)$ made by V^*, Sim records u in a set L. To simulate the first message, Sim invokes the simulator of Π_{OT} for the corrupt receiver. Upon receiving V^*'s message c, Sim parses c as $(\mathbf{C}, \{m_j^S\}_{j\in[n]}, T)$ and defines the set $\tilde{R} = \{H(u) \oplus T | u \in L\}$. For any $r \in \tilde{R}$ parsed as $r = r_1 || \ldots || r_n$, Sim computes $(k_j^0, k_j^1) \leftarrow \mathsf{Ext}(m_j^R, m_j^S, r_j^S)$ for $j \in [n]$ and checks if $\mathsf{Verify}(\hat{C}, \mathbf{C}, \{k_j^0, k_j^1\}_{j\in[n]}) = 1$. If there exists such $r \in \tilde{R}$, the simulator sends Y to V^*, where $Y \in L$ is so that $r = H(Y) \oplus T$. Otherwise, Sim aborts the protocol. It should be clear that the output of the simulator is perfectly indistinguishable from the real distribution. This completes the proof. □

6.6 Applications: Witness Encryption with Decryptor Privacy

Besides being a notion of theoretical interest, we also show the applications of (commit-and-prove) predictable arguments with zero-knowledge or witness-indistinguishability property in the context of witness encryption. Witness encryption (WE) is a powerful notion of encryption introduced by Garg et al. [98]. A WE scheme for an NP relation $\mathcal{R}_{\mathsf{lpar}}$ allows to encrypt a message m with respect to a statement x as $ct \leftarrow \mathsf{WE.Enc}(\mathsf{lpar}, m, \mathsf{x})$. The ciphertext can be decrypted as $m \leftarrow \mathsf{WE.Dec}(ct, \mathsf{w})$ for any w such that $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathsf{lpar}}$. Security guarantees that no adversary should learn any non-trivial information about m if $\mathsf{x} \notin \mathcal{L}_{\mathsf{lpar}}$, where $\mathcal{L}_{\mathsf{lpar}}$ is the language corresponding to $\mathcal{R}_{\mathsf{lpar}}$. More formally, we say that a WE is secure if it is complete and sound as defined below:

- **Completeness.** A WE has completeness if for all $\lambda \in \mathbb{N}$, for all $\mathcal{R}_{\mathsf{lpar}} \in \mathcal{RG}_\lambda$, for all m,

- **Oracles and Primitives**: A correct, authentic, and verifiable garbling scheme $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode})$, a sender-extractable 2-round OT Π_{OT}, and a hash function $H : \{0,1\}^* \to \{0,1\}^{\mathsf{poly}(\lambda)}$ modeled as a random oracle.
- P**'s private input**: $\mathsf{w} \in \{0,1\}^n$, where $n = \mathsf{poly}(\lambda)$.
- **Commitment Phase**: P plays the role of the receiver in n instances of Π_{OT} and computes $(\mathsf{cm}, \mathsf{d})$ as follows:
  1. Sample uniformly random r_j^R from $\{0,1\}^\lambda$, and compute $m_j^R \leftarrow \Pi_{\mathsf{OT}}^R(\mathsf{w}_j; r_j^R)$ for $j \in [n]$.
  2. Define $\mathsf{cm} = \{m_j^R\}_{j\in[n]}$ and $\mathsf{d} = \{r_j^R\}_{j\in[n]}$.
- **Common inputs**: A security parameter λ, and a statement $\mathsf{x} = (\mathsf{cm}, C, y)$, where C is a polynomial-size circuit.
- **Challenge**: Let $\hat{C}$ be a circuit that realizes the following relation $\mathcal{R}$: $\mathcal{R}(\mathsf{x} = (\mathsf{cm}, C, y), (\mathsf{w}, \mathsf{d})) = 1$ iff (w, d) opens cm and $C(\mathsf{w}) = y$. V plays the role of the sender in n instances of Π_{OT} and computes a pair (c, b) of challenge-predicted answer as follows:
  1. Compute $(\mathbf{C}, e, d) \leftarrow \mathsf{Garble}(1^\lambda, \hat{C})$, where $e := \{k_j^0, k_j^1\}_{j\in[n]}$, and $d := (k^0, k^1)$.
  2. For $j \in [n]$, sample uniformly random r_j^S from $\{0,1\}^\lambda$, and compute $m_j^S = \Pi_{\mathsf{OT}}^S(k_j^0, k_j^1, m_j^R; r_j^S)$.
  3. Compute $T = H(k^1) \oplus r^S$, where $r^S = r_1^S || \ldots || r_n^S$.
  4. Define $c = (\mathbf{C}, \{m_j^S\}_{j\in[n]}, T)$ and $b = k^1$, and send c to P.
- **Response**: P proceeds as follows:
  1. Execute $k_j^{\mathsf{w}_j} = \Pi_{\mathsf{OT}}^O(m_j^S, \mathsf{w}_j, r_j^R)$ for $j \in [n]$.
  2. Execute $Y = \mathsf{Eval}(\mathbf{C}, \{k_j^{\mathsf{w}_j}\}_{j\in[n]})$.
  3. Recover $r^S = H(Y) \oplus T$, and parse $r^S = r_1^S || \ldots || r_n^S$.
  4. Reconstruct sender's inputs $(k_j^0, k_j^1) \leftarrow \mathsf{Ext}(m_j^R, m_j^S, r_j^S)$ for $j \in [n]$. Abort if the extractor fails for some $j \in [n]$.
  5. Send the predicted answer $a = Y$ if $\mathsf{Verify}(\hat{C}, \mathbf{C}, \{k_j^0, k_j^1\}_{j\in[n]}) = 1$; and abort otherwise.
- V accepts the proof iff $a = b$.

Figure 6.5: ZK-CPPA Π_{cppa} based on GC and OT

and for all $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathsf{lpar}}$

$$\Pr[\mathsf{Dec}(\mathsf{Enc}(\mathsf{lpar}, \mathsf{x}, m), \mathsf{w}) = m] \geq 1 - \mathsf{negl}(\lambda)$$

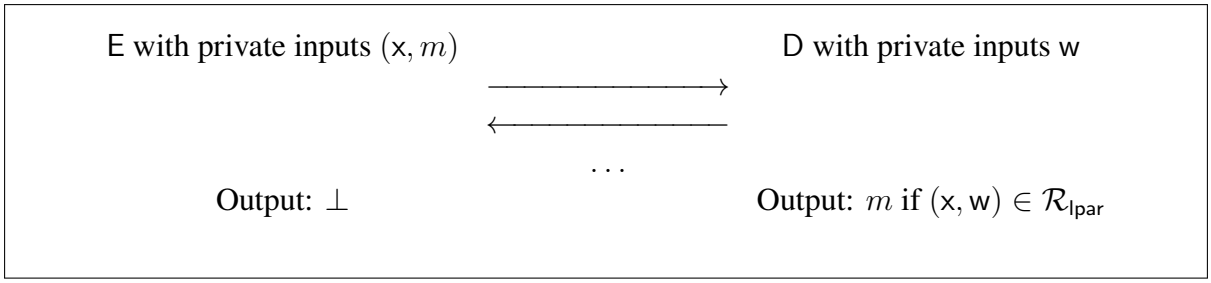

Figure 6.6: Functionality of a WE scheme with decryptor privacy for a relation $\mathcal{R}_{\mathsf{lpar}}$

If the probability is 1, we say WE is perfectly complete.

- **Soundness.** A WE has soundness if for all $\lambda \in \mathbb{N}$ and all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that for any m_0, m_1

$$\Pr\left[\begin{matrix}\mathcal{R}_{\mathsf{lpar}} \leftarrow\$ \mathcal{RG}_\lambda; \mathsf{x} \leftarrow \mathcal{A}(\mathsf{lpar}); b \leftarrow\$ \{0,1\}; \\ ct \leftarrow \mathsf{Enc}(\mathsf{lpar}, \mathsf{x}, m_b); b' \leftarrow \mathcal{A}(\mathsf{lpar}, \mathsf{x}, ct)\end{matrix} : \begin{matrix} b = b' \wedge \mathsf{x} \notin \mathcal{L}_{\mathsf{lpar}} \\ \wedge |m_0| = |m_1| \end{matrix}\right] \approx_\lambda \mathsf{negl}(\lambda)$$

While being a very powerful notion, existing constructions of WE are not satisfactory, as they are either based on strong assumptions such as indistinguishability obfuscation and multilinear maps [61, 97, 98, 111], or based on new and unexplored algebraic structures [18].

As noted in [79], predictable arguments imply witness encryption as one can encrypt a bit m by generating a challenge-answer pair (c, b) for the PA and define the ciphertext as $(c, b \oplus m)$. Viceversa, a PA can be constructed from WE by encrypting a random bit m and then asking the prover to return m. Furthermore, it is not hard to show that commit-and-prove predictable arguments are also equivalent to a variant of witness encryption studied in [29, 49]. It is therefore interesting to see the applications of predictable arguments with privacy in the context of witness encryption. While the standard definition of witness encryption requires the above properties, for some applications explained below, we may require some level of privacy for the decryptor as well. In other words, we may ask for a WE scheme that mimics the following functionality (See Fig. 6.6): the functionality is parameterized by a message space $\mathcal{M}$ and an NP relation $\mathcal{R}_{\mathsf{lpar}}$. An encryptor E with private inputs $m \in \mathcal{M}$ and bitstring x interacts with a decryptor D with private input w, at the end of which D outputs m iff $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$. Note that this is different from standard WE wherein the decryptor aims to obtain the message internally without revealing it to the environment. Here instead, the decrypted message is revealed to the encryptor which may break the privacy of the decryptor.

Since the encryptor knows the plaintext when running the encryption algorithm, one may wonder how the decrypted message can leak some information about the decryptor's witness. To clarify this, let us consider a WE scheme for a concrete disjunction language defined as follows: the language parameter $\mathsf{lpar} = (\mathbb{G}, g, \mathsf{pk})$ includes a group $\mathbb{G}$ of order p with generator g, and an ElGamal public key pk. A statement x is in the language iff x is the ElGamal encryption of a bit under pk. More formally, for $\mathsf{lpar} = (\mathbb{G}, g, \mathsf{pk})$, we define

$$\mathcal{L}_{\mathsf{lpar}} = \left\{\mathsf{x} \mid \exists r \in \mathbb{Z}_p, \exists b \in \{0,1\} : \mathsf{x} = (g^r, \mathsf{pk}^r g^b)\right\}$$

We denote the witness for $\mathsf{x} = (g^r, \mathsf{pk}^r g^b)$ as $\mathsf{w} = (r, b)$. Using generic techniques for the disjunctions of languages([2, 121]), one can encrypt a message $m \in \mathbb{G}$ under a statement $\mathsf{x} = (\mathsf{x}_0, \mathsf{x}_1)$ as follows:

1. select $\alpha_0, \alpha_1, \alpha_2, \alpha_3 \leftarrow\$ \mathbb{Z}_p$.
2. compute $\mathsf{aux}_0 = g^{\alpha_0}\mathsf{pk}^{\alpha_1}$, $\mathsf{aux}_1 = g^{\alpha_1}\mathsf{x}_0^{\alpha_2}(\frac{\mathsf{x}_1}{g})^{\alpha_3}$ and $\mathsf{aux}_2 = g^{\alpha_2}\mathsf{pk}^{\alpha_3}$. Define $\mathsf{aux} = (\mathsf{aux}_0, \mathsf{aux}_1, \mathsf{aux}_2)$.
3. compute $\pi = \mathsf{x}_0^{\alpha_0}\mathsf{x}_1^{\alpha_1}$ and $\tilde{\pi} = \pi\mathsf{x}_0^{\alpha_0}(\frac{\mathsf{x}_1}{g})^{\alpha_1}g^{\alpha_1}$. Define $ct = \pi m$.
4. return $ct = (c, \mathsf{aux}, \tilde{\pi})$.

Having a witness $\mathsf{w} = (r, 0)$ for $\mathsf{x} = (g^r, \mathsf{pk}^r)$, one can decrypt the ciphertext by first computing $\pi = \mathsf{aux}_0^r$ and then recovering the message by computing $\pi^{-1}ct$. On the other hand, if the decryptor has a witness $\mathsf{w} = (r, 1)$ for $\mathsf{x} = (g^r, \mathsf{pk}^r g)$, he first obtains π from dividing $\tilde{\pi}$ by $\mathsf{aux}_0^r\mathsf{aux}_1^1\mathsf{aux}_2^{-r} = \mathsf{x}_0^{\alpha_0}(\frac{\mathsf{x}_1}{g})^{\alpha_1}g^{\alpha_1}$ and then computes $m = \pi^{-1}ct$ as before. While for an honestly generated ciphertext ct, this construction does not leak any information about the witness, it is not hard to see that a malicious encryptor can learn part of the witness (and thus distinguish them) by simply defining $\tilde{\pi}$ to be a random group element. In this case (i.e., $b = 1$), the decryptor fails to decrypt m correctly, hence making the two cases $b = 0$ and $b = 1$ distinguishable for the encryptor.

6.6.1 Application: Dark Pools

We now justify our model of WE with decryptor privacy. In our model, we are assuming that the decryptor D sends back the decrypted message to the encryptor E whereas in all previous works, the communication is non-interactive (i.e., "one-shot") in the sense that there is only one message ct from E to D. Our motivating applications are *dark pools* and *over-the-counter* markets. Dark pools are anonymized trading platforms that allow parties to place invisible orders such that each party can only know their own orders. Such pools allow the investors to communicate only to those whose transaction conditions satisfy some constraints. At the same time, they should also guarantee that investors do not learn any information about traders' secret information.

In a recent work, Ngo et al. [138] introduced a new cryptographic primitive called *Witness Key Agreement* (WKA) as a tool to make this possible. In the dark pool scenario, a WKA allows a party E to securely agree on a secret key with another party D who owns a secret witness satisfying some arithmetic relation. More precisely, in the presence of a public bulletin board or a public blockchain, a WKA addresses the following problem: given n parties who have committed to their secret inputs w, and published the commitments cm anonymously on the blockchain, an investor E wants to agree on a key k with any party whose committed secret w satisfies some relation; i.e., $C(\mathsf{w}) = y$, where C is an arbitrary arithmetic circuit specified by E. Similar to NP relations defined in Section 6.5, one can set $\mathsf{x} = (\mathsf{cm}, C, y)$ and let $\mathcal{R}$ be defined such that $\mathcal{R}(\mathsf{x}, (\mathsf{w}, \mathsf{d})) = 1$ iff cm commits to w (with decommitment d) and $C(\mathsf{w}) = y$. Once the secret key k is recovered by the legitimate party (i.e., any party with valid witness (w, d) such that $\mathcal{R}(\mathsf{x}, (\mathsf{d}, \mathsf{w})) = 1$), they together with the investor can secure their communication from any external party by using k.

We now demonstrate how our construction of ZK-CPPA can be used as a drop-in replacement for a witness key agreement. At a high level, the protocol proceeds as follows. All parties first commit to their secret values w via $\mathsf{cm} \leftarrow \mathsf{Commit}(\mathsf{w}; \mathsf{d})$, and publish the resulting commitments cm. Later, an investor who wish to communicate only with participants whose secret satisfy $C(\mathsf{w}) = y$ (for some arbitrarily chosen circuit C and value y) considers the following relation:

$\mathcal{R}(\mathsf{x} = (\mathsf{cm}, C, y), (\mathsf{w}, \mathsf{d})) = 1$ iff $C(\mathsf{w}) = y$, and $\mathsf{cm} = \mathsf{Commit}(\mathsf{w}; \mathsf{d})$. Let us assume that $\mathsf{x}_i = (\mathsf{cm}_i, C, y)$ is the statement corresponding to party i. The investor now encrypts the secret key k under all such statements x_i [5]. It is not hard to see that only the prover with the valid witness $(\mathsf{w}_i, \mathsf{d}_i)$ can decrypt the ciphertext. Moreover, since the construction is ZK, the decrypted message k says nothing about $(\mathsf{w}_i, \mathsf{d}_i)$, even if the ciphertext is generated maliciously.

Efficiency and Comparison with [138]. In [138], the authors propose a WKA construction based on a type of *Succinct Zero-Knowledge Non-Interactive Argument of Knowledge Proof System* (zk-SNARK) from non-interactive linear proof systems (NILP), where the verifier is designated. The construction at a high-level is as follows. A designated verifier—playing the role of the investor— first broadcasts a CRS as a challenge for the relation $\mathcal{R}$ of interest. Next, a prover publishes a partial zk-SNARK proof as a response for the committed value that satisfies $\mathcal{R}$. Finally, the verifier using the partial proof can derive a shared secret key with the prover.

We now compare our proposed construction for WKA with that of [138]. In contrast to our scheme which is ZK, the construction of [138] only provides honest-verifier ZK. Moreover, the WKA in [138] requires an expensive trusted setup which should be invoked every time an investor E_i asks for the preprocessing of a new CRS corresponding to the relation $\mathcal{R}_i$ of E_i's interest. On the other hand, the major downside of our scheme is that the size of the ciphertext grows linearly with the number of parties in the system as the investor should encrypt the message under every existing commitment in the system, whereas the size of ciphertext in [138] is independent of the number of parties. This suggests that there might well be a trade-off between the size of the ciphertext and the required number of trusted setups and our construction performs better when the number of parties is small.

[5] We again emphasize that we see the notions of PA and WE (and their "commit-and-prove" variants) interchangeably here, as the implication from one to another is straightforward and shown in [79].

Bibliography

[1] Zcash regulatory and compliance brief. https://z.cash/wp-content/uploads/2020/07/Zcash-Regulatory-Brief-062020.pdf. 2020-06-01.

[2] Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth projective hashing for conditionally extractable commitments. pages 671–689, 2009. doi: 10.1007/978-3-642-03356-8_39.

[3] Behzad Abdolmaleki, Sebastian Ramacher, and Daniel Slamanig. Lift-and-shift: Obtaining simulation extractable subversion and updatable SNARKs generically.

[4] Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, and Michal Zajac. A subversion-resistant SNARK. pages 3–33, 2017. doi: 10.1007/978-3-319-70700-6_1.

[5] Behzad Abdolmaleki, Sebastian Ramacher, and Daniel Slamanig. Lift-and-shift: Obtaining simulation extractable subversion and updatable SNARKs generically. pages 1987–2005, 2020. doi: 10.1145/3372297.3417228.

[6] Behzad Abdolmaleki, Hamidreza Khoshakhlagh, and Helger Lipmaa. Smooth zero-knowledge hash functions. Cryptology ePrint Archive, Report 2021/653, 2021. https://eprint.iacr.org/2021/653.

[7] Behzad Abdolmaleki, Hamidreza Khoshakhlagh, and Helger Lipmaa. Smooth zero-knowledge hash functions. In *Progress in Cryptology - INDOCRYPT 2021 - 22nd International Conference on Cryptology in India, Jaipur, India, December 12-15, 2021, Proceedings*, pages 510–535, 2021. doi: 10.1007/978-3-030-92518-5_23. URL https://doi.org/10.1007/978-3-030-92518-5_23.

[8] Arash Afshar, Payman Mohassel, Benny Pinkas, and Ben Riva. Non-interactive secure computation based on cut-and-choose. pages 387–404, 2014. doi: 10.1007/978-3-642-55220-5_22.

[9] Shashank Agrawal, Chaya Ganesh, and Payman Mohassel. Non-interactive zero-knowledge proofs for composite statements. pages 643–673, 2018. doi: 10.1007/978-3-319-96878-0_22.

[10] Elli Androulaki, Jan Camenisch, Angelo De Caro, Maria Dubovitskaya, Kaoutar Elkhiyaoui, and Björn Tackmann. Privacy-preserving auditable token payments in a permissioned blockchain system. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 255–267, 2020.

[11] Shahla Atapoor and Karim Baghery. Simulation extractability in groth's zk-SNARK. Cryptology ePrint Archive, Report 2019/641, 2019. https://eprint.iacr.org/2019/641.

[12] Thomas Attema, Serge Fehr, and Michael Klooß. Fiat-shamir transformation of multi-round interactive proofs. Cryptology ePrint Archive, Report 2021/1377, 2021. https://eprint.iacr.org/2021/1377.

[13] Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic k-TAA. pages 111–125, 2006. doi: 10.1007/11832072_8.

[14] Man Ho Au, Willy Susilo, Yi Mu, and Sherman SM Chow. Constant-size dynamic k-times anonymous authentication. *IEEE Systems Journal*, 7(2):249–261, 2012.

[15] Aztec. A private layer 2. https://developers.aztec.network, 2020.

[16] Karim Baghery, Markulf Kohlweiss, Janno Siim, and Mikhail Volkhov. Another look at extraction and randomization of groth's zk-SNARK. Cryptology ePrint Archive, Report 2020/811, 2020. https://eprint.iacr.org/2020/811.

[17] Ohad Barta, Yuval Ishai, Rafail Ostrovsky, and David J. Wu. On succinct arguments and witness encryption from groups. pages 776–806, 2020. doi: 10.1007/978-3-030-56784-2_26.

[18] James Bartusek, Yuval Ishai, Aayush Jain, Fermi Ma, Amit Sahai, and Mark Zhandry. Affine determinant programs: A framework for obfuscation and witness encryption. pages 82:1–82:39, 2020. doi: 10.4230/LIPIcs.ITCS.2020.82.

[19] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. pages 390–399, 2006. doi: 10.1145/1180405.1180453.

[20] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. pages 62–73, 1993. doi: 10.1145/168588.168596.

[21] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. pages 784–796, 2012. doi: 10.1145/2382196.2382279.

[22] Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. pages 777–804, 2016. doi: 10.1007/978-3-662-53890-6_26.

[23] Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. pages 37–56, 1990. doi: 10.1007/0-387-34799-2_4.

[24] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. pages 90–108, 2013. doi: 10.1007/978-3-642-40084-1_6.

[25] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. pages 459–474, 2014. doi: 10.1109/SP.2014.36.

[26] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. pages 781–796, 2014.

[27] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. pages 31–60, 2016. doi: 10.1007/978-3-662-53644-5_2.

[28] 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.

[29] Fabrice Benhamouda and Huijia Lin. Mr NISC: Multiparty reusable non-interactive secure computation. pages 349–378, 2020. doi: 10.1007/978-3-030-64378-2_13.

[30] Fabrice Benhamouda and David Pointcheval. Trapdoor smooth projective hash functions. Cryptology ePrint Archive, Report 2013/341, 2013. https://eprint.iacr.org/2013/341.

[31] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. pages 449–475, 2013. doi: 10.1007/978-3-642-40041-4_25.

[32] Fabrice Benhamouda, Craig Gentry, Sergey Gorbunov, Shai Halevi, Hugo Krawczyk, Chengyu Lin, Tal Rabin, and Leonid Reyzin. Can a public blockchain keep a secret? Cryptology ePrint Archive, Report 2020/464, 2020. https://eprint.iacr.org/2020/464.

[33] Fabrice Benhamouda, Craig Gentry, Sergey Gorbunov, Shai Halevi, Hugo Krawczyk, Chengyu Lin, Tal Rabin, and Leonid Reyzin. Can a public blockchain keep a secret? pages 260–290, 2020. doi: 10.1007/978-3-030-64375-1_10.

[34] Nir Bitansky and Arka Rai Choudhuri. Characterizing deterministic-prover zero knowledge. pages 535–566, 2020. doi: 10.1007/978-3-030-64375-1_19.

[35] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. pages 315–333, 2013. doi: 10.1007/978-3-642-36594-2_18.

[36] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. pages 505–514, 2014. doi: 10.1145/2591796.2591859.

[37] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. pages 213–229, 2001. doi: 10.1007/3-540-44647-8_13.

[38] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. pages 327–357, 2016. doi: 10.1007/978-3-662-49896-5_12.

[39] Sean Bowe and Ariel Gabizon. Making groth's zk-SNARK simulation extractable in the random oracle model. Cryptology ePrint Archive, Report 2018/187, 2018. https://eprint.iacr.org/2018/187.

[40] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of computer and system sciences*, 37(2):156–189, 1988.

[41] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. pages 315–334, 2018. doi: 10.1109/SP.2018.00020.

[42] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). pages 410–424, 1997. doi: 10.1007/BFb0052252.

[43] Jan Camenisch, Ueli M. Maurer, and Markus Stadler. Digital payment systems with passive anonymity-revoking trustees. pages 33–43, 1996. doi: 10.1007/3-540-61770-1_26.

[44] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. pages 302–321, 2005. doi: 10.1007/11426639_18.

[45] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: Efficient periodic n-times anonymous authentication. pages 201–210, 2006. doi: 10.1145/1180405.1180431.

[46] Matteo Campanelli and Hamidreza Khoshakhlagh. Succinct publicly-certifiable proofs (or: Can a blockchain verify a designated-verifier proof?). Cryptology ePrint Archive, Report 2021/1618, 2021. https://eprint.iacr.org/2021/1618.

[47] Matteo Campanelli and Hamidreza Khoshakhlagh. Succinct publicly-certifiable proofs - or, can a blockchain verify a designated-verifier proof? In *Progress in Cryptology - INDOCRYPT 2021 - 22nd International Conference on Cryptology in India, Jaipur, India, December 12-15, 2021, Proceedings*, pages 607–631, 2021. doi: 10.1007/978-3-030-92518-5_27. URL https://doi.org/10.1007/978-3-030-92518-5_27.

[48] Matteo Campanelli, Dario Fiore, and Anaïs Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. pages 2075–2092, 2019. doi: 10.1145/3319535.3339820.

[49] Matteo Campanelli, Bernardo David, Hamidreza Khoshakhlagh, Anders Konring, and Jesper Buus Nielsen. Encryption to the future: A paradigm for sending secret messages to future (anonymous) committees. Cryptology ePrint Archive, Report 2021/1423, 2021. https://eprint.iacr.org/2021/1423.

[50] Matteo Campanelli, Chaya Ganesh, Hamidreza Khoshakhlagh, and Janno Siim. Impossibilities in succinct arguments: Black-box extraction and more. Cryptology ePrint Archive, Report 2022/638, 2022. https://eprint.iacr.org/2022/638.

[51] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. https://eprint.iacr.org/2000/067.

[52] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. pages 136–145, 2001. doi: 10.1109/SFCS.2001.959888.

[53] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. pages 494–503, 2002. doi: 10.1145/509907.509980.

[54] Guilhem Castagnos and Fabien Laguillaumie. Linearly homomorphic encryption from DDH. pages 487–505, 2015. doi: 10.1007/978-3-319-16715-2_26.

[55] Guilhem Castagnos, Fabien Laguillaumie, and Ida Tucker. Practical fully secure unrestricted inner product functional encryption modulo p. pages 733–764, 2018. doi: 10.1007/978-3-030-03329-3_25.

[56] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Bandwidth-efficient threshold EC-DSA. pages 266–296, 2020. doi: 10.1007/978-3-030-45388-6_10.

[57] Ethan Cecchetti, Fan Zhang, Yan Ji, Ahmed E. Kosba, Ari Juels, and Elaine Shi. Solidus: Confidential distributed ledger transactions via PVORM. pages 701–717, 2017. doi: 10.1145/3133956.3134010.

[58] Suvradip Chakraborty, Manoj Prabhakaran, and Daniel Wichs. Witness maps and applications. pages 220–246, 2020. doi: 10.1007/978-3-030-45374-9_8.

[59] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. pages 78–96, 2006. doi: 10.1007/11818175_5.

[60] David Chaum. Blind signature system. page 153, 1983.

[61] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. pages 577–607, 2018. doi: 10.1007/978-3-319-96881-0_20.

[62] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. pages 738–768, 2020. doi: 10.1007/978-3-030-45721-1_26.

[63] Sherman S. M. Chow. Real traceable signatures. pages 92–107, 2009. doi: 10.1007/978-3-642-05445-7_6.

[64] cLabs. The celo protocol: A multi-asset cryptographic protocol fordecentralized social payments. https://celo.org/papers/whitepaper, 2020.

[65] Clearmatics. Zeth: On integrating zerocash on ethereum. https://www.github.com/clearmatics/zeth, 2020.

[66] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. pages 45–64, 2002. doi: 10.1007/3-540-46035-7_4.

[67] Hila Dahari and Yehuda Lindell. Deterministic-prover zero-knowledge proofs. Cryptology ePrint Archive, Report 2020/141, 2020. https://eprint.iacr.org/2020/141.

[68] Anders P. K. Dalskov, Claudio Orlandi, Marcel Keller, Kris Shrishak, and Haya Shulman. Securing DNSSEC keys via threshold ECDSA from generic MPC. pages 654–673, 2020. doi: 10.1007/978-3-030-59013-0_32.

[69] Ivan Damgård, Kasper Dupont, and Michael Østergaard Pedersen. Unclonable group identification. pages 555–572, 2006. doi: 10.1007/11761679_33.

[70] Ivan Damgård, Chaya Ganesh, Hamidreza Khoshakhlagh, Claudio Orlandi, and Luisa Siniscalchi. Balancing privacy and accountability in blockchain identity management. Cryptology ePrint Archive, Report 2020/1511, 2020. https://eprint.iacr.org/2020/1511.

[71] Ivan Damgård, Chaya Ganesh, Hamidreza Khoshakhlagh, Claudio Orlandi, and Luisa Siniscalchi. Balancing privacy and accountability in blockchain identity management. pages 552–576, 2021. doi: 10.1007/978-3-030-75539-3_23.

[72] George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. pages 532–550, 2014. doi: 10.1007/978-3-662-45611-8_28.

[73] Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. pages 66–98, 2018. doi: 10.1007/978-3-319-78375-8_3.

[74] Vanesa Daza, Javier Herranz, Paz Morillo, and Carla Ràfols. CCA2-secure threshold broadcast encryption with shorter ciphertexts. pages 35–50, 2007.

[75] Alfredo De Santis and Giuseppe Persiano. Zero-knowledge proofs of knowledge without interaction (extended abstract). pages 427–436, 1992. doi: 10.1109/SFCS.1992.267809.

[76] David Derler and Daniel Slamanig. Practical witness encryption for algebraic languages and how to reply an unknown whistleblower. Cryptology ePrint Archive, Report 2015/1073, 2015. https://eprint.iacr.org/2015/1073.

[77] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. pages 416–431, 2005. doi: 10.1007/978-3-540-30580-4_28.

[78] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. pages 613–631, 2010. doi: 10.1007/978-3-642-17373-8_35.

[79] Antonio Faonio, Jesper Buus Nielsen, and Daniele Venturi. Predictable arguments of knowledge. pages 121–150, 2017. doi: 10.1007/978-3-662-54365-8_6.

[80] Pooya Farshim, Claudio Orlandi, and Răzvan Roşie. Security of symmetric primitives under incorrect usage of keys. 2017(1):449–473, 2017. doi: 10.13154/tosc.v2017.i1.449-473.

[81] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the Fiat-Shamir transform. pages 60–79, 2012. doi: 10.1007/978-3-642-34931-7_5.

[82] Prastudy Fauzi, Sarah Meiklejohn, Rebekah Mercer, and Claudio Orlandi. Quisquis: A new design for anonymous cryptocurrencies. pages 649–678, 2019. doi: 10.1007/978-3-030-34578-5_23.

[83] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. pages 416–426, 1990. doi: 10.1145/100216.100272.

[84] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. pages 186–194, 1987. doi: 10.1007/3-540-47721-7_12.

[85] Dario Fiore and Anca Nitulescu. On the (in)security of SNARKs in the presence of oracles. pages 108–138, 2016. doi: 10.1007/978-3-662-53641-4_5.

[86] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. pages 152–168, 2005. doi: 10.1007/11535218_10.

[87] Lance Fortnow. The complexity of perfect zero-knowledge (extended abstract). pages 204–209, 1987. doi: 10.1145/28395.28418.

[88] Georg Fuchsbauer. Subversion-zero-knowledge SNARKs. pages 315–347, 2018. doi: 10.1007/978-3-319-76578-5_11.

[89] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. pages 33–62, 2018. doi: 10.1007/978-3-319-96881-0_2.

[90] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. pages 537–554, 1999. doi: 10.1007/3-540-48405-1_34.

[91] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. https://eprint.iacr.org/2019/953.

[92] Chaya Ganesh, Yashvanth Kondi, Arpita Patra, and Pratik Sarkar. Efficient adaptively secure zero-knowledge from garbled circuits. pages 499–529, 2018. doi: 10.1007/978-3-319-76581-5_17.

[93] Chaya Ganesh, Hamidreza Khoshakhlagh, Markulf Kohlweiss, Anca Nitulescu, and Michal Zajac. What makes fiat–shamir zksnarks (updatable srs) simulation extractable? Cryptology ePrint Archive, Report 2021/511, 2021. https://eprint.iacr.org/2021/511.

[94] Chaya Ganesh, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. Fiat–shamir bulletproofs are non-malleable (in the algebraic group model). Cryptology ePrint Archive, Report 2021/1393, 2021. https://ia.cr/2021/1393.

[95] Juan A. Garay, Ran Gelles, David S. Johnson, Aggelos Kiayias, and Moti Yung. A little honesty goes a long way - the two-tier model for secure multiparty computation. pages 134–158, 2015. doi: 10.1007/978-3-662-46494-6_7.

[96] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. pages 281–310, 2015. doi: 10.1007/978-3-662-46803-6_10.

[97] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. pages 40–49, 2013. doi: 10.1109/FOCS.2013.13.

[98] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. pages 467–476, 2013. doi: 10.1145/2488608.2488667.

[99] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. pages 626–645, 2013. doi: 10.1007/978-3-642-38348-9_37.

[100] Craig Gentry, Allison B. Lewko, and Brent Waters. Witness encryption from instance independent assumptions. pages 426–443, 2014. doi: 10.1007/978-3-662-44371-2_24.

[101] Craig Gentry, Shai Halevi, Bernardo Magri, Jesper Buus Nielsen, and Sophia Yakoubov. Random-index PIR with applications to large-scale secure MPC. Cryptology ePrint Archive, Report 2020/1248, 2020. https://eprint.iacr.org/2020/1248.

[102] Craig Gentry, Shai Halevi, Hugo Krawczyk, Bernardo Magri, Jesper Buus Nielsen, Tal Rabin, and Sophia Yakoubov. YOSO: You only speak once - secure MPC with stateless ephemeral roles. pages 64–93, 2021. doi: 10.1007/978-3-030-84245-1_3.

[103] Craig Gentry, Shai Halevi, Bernardo Magri, Jesper Buus Nielsen, and Sophia Yakoubov. Random-index pir and applications. In *Theory of Cryptography Conference*, pages 32–61. Springer, 2021.

[104] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. Cryptology ePrint Archive, Report 2017/454, 2017. https://eprint.iacr.org/2017/454.

[105] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. 7(1):1–32, December 1994. doi: 10.1007/BF00195207.

[106] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). pages 174–187, 1986. doi: 10.1109/SFCS.1986.47.

[107] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. pages 102–115, 2003. doi: 10.1109/SFCS.2003.1238185.

[108] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. 17(2):281–308, April 1988.

[109] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. 18(1):186–208, 1989.

[110] Rishab Goyal and Vipul Goyal. Overcoming cryptographic impossibility results using blockchains. pages 529–561, 2017. doi: 10.1007/978-3-319-70500-2_18.

[111] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. pages 612–621, 2017. doi: 10.1109/FOCS.2017.62.

[112] Vipul Goyal, Abhiram Kothapalli, Elisaweta Masserova, Bryan Parno, and Yifan Song. Storing and retrieving secrets on a blockchain. Cryptology ePrint Archive, Report 2020/504, 2020. https://eprint.iacr.org/2020/504.

[113] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. pages 444–459, 2006. doi: 10.1007/11935230_29.

[114] Jens Groth. Fully anonymous group signatures without random oracles. pages 164–180, 2007. doi: 10.1007/978-3-540-76900-2_10.

[115] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. pages 321–340, 2010. doi: 10.1007/978-3-642-17373-8_19.

[116] Jens Groth. On the size of pairing-based non-interactive arguments. pages 305–326, 2016. doi: 10.1007/978-3-662-49896-5_11.

[117] Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. pages 581–612, 2017. doi: 10.1007/978-3-319-63715-0_20.

[118] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. pages 97–111, 2006. doi: 10.1007/11818175_6.

[119] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11:1–11:35, 2012. doi: 10.1145/2220357.2220358. URL https://doi.org/10.1145/2220357.2220358.

[120] Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. pages 698–728, 2018. doi: 10.1007/978-3-319-96878-0_24.

[121] Fabrice Ben Hamouda-Guichoux. *Diverse modules and zero-knowledge*. PhD thesis, École Normale Supérieure, Paris, France, 2016. URL https://tel.archives-ouvertes.fr/tel-01399476.

[122] Marek Jawurek, Florian Kerschbaum, and Claudio Orlandi. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. pages 955–966, 2013. doi: 10.1145/2508859.2516662.

[123] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. pages 177–194, 2010. doi: 10.1007/978-3-642-17373-8_11.

[124] Hamidreza Khoshakhlagh. (commit-and-prove) predictable arguments with privacy. *IACR Cryptol. ePrint Arch.*, page 377, 2022. URL https://eprint.iacr.org/2022/377.

[125] Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Traceable signatures. pages 571–589, 2004. doi: 10.1007/978-3-540-24676-3_34.

[126] Aggelos Kiayias, Hong-Sheng Zhou, and Vassilis Zikas. Fair and robust multi-party computation using a global transaction ledger. pages 705–734, 2016. doi: 10.1007/978-3-662-49896-5_25.

[127] Ahmed Kosba, Zhichao Zhao, Andrew Miller, Yi Qian, Hubert Chan, Charalampos Papamanthou, Rafael Pass, abhi shelat, and Elaine Shi. How to use SNARKs in universally composable protocols. Cryptology ePrint Archive, Report 2015/1093, 2015. https://eprint.iacr.org/2015/1093.

[128] Ahmed E. Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. pages 839–858, 2016. doi: 10.1109/SP.2016.55.

[129] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. pages 169–189, 2012. doi: 10.1007/978-3-642-28914-9_10.

[130] Helger Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. pages 41–60, 2013. doi: 10.1007/978-3-642-42033-7_3.

[131] Helger Lipmaa. Key-and-argument-updatable QA-NIZKs. Cryptology ePrint Archive, Report 2019/333, 2019. https://eprint.iacr.org/2019/333.

[132] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. pages 2111–2128, 2019. doi: 10.1145/3319535.3339817.

[133] Ueli M. Maurer. Unifying zero-knowledge proofs of knowledge. pages 272–286, 2009.

[134] 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 *Proceedings of the 2013 conference on Internet measurement conference*, pages 127–140, 2013.

[135] Silvio Micali. CS proofs (extended abstracts). pages 436–453, 1994. doi: 10.1109/SFCS.1994.365746.

[136] Jim Miller. Coordinated disclosure of vulnerabilities affecting girault, bulletproofs, and plonk. https://blog.trailofbits.com/2022/04/13/part-1-coordinated-disclosure-of-vulnerabilities-affecting-girault-bulletproof 2022.

[137] Neha Narula, Willy Vasquez, and Madars Virza. zkLedger: Privacy-preserving auditing for distributed ledgers. Cryptology ePrint Archive, Report 2018/241, 2018. https://eprint.iacr.org/2018/241.

[138] Chan Nam Ngo, Fabio Massacci, Florian Kerschbaum, and Julian Williams. Practical witness-key-agreement for blockchain-based dark pools financial trading. In *Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part II*, pages 579–598, 2021. doi: 10.1007/978-3-662-64331-0_30. URL https://doi.org/10.1007/978-3-662-64331-0_30.

[139] Lan Nguyen and Reihaneh Safavi-Naini. Dynamic k-times anonymous authentication. pages 318–333, 2005. doi: 10.1007/11496137_22.

[140] Jesper Buus Nielsen. *On protocol security in the cryptographic model*. Citeseer, 2003.

[141] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. pages 238–252, 2013. doi: 10.1109/SP.2013.47.

[142] Rafael Pass, Lior Seeman, and abhi shelat. Analysis of the blockchain protocol in asynchronous networks. pages 643–673, 2017. doi: 10.1007/978-3-319-56614-6_22.

[143] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. pages 129–140, 1992. doi: 10.1007/3-540-46766-1_9.

[144] Andrew Poelstra, Adam Back, Mark Friedenbach, Gregory Maxwell, and Pieter Wuille. Confidential assets. pages 43–63, 2019. doi: 10.1007/978-3-662-58820-8_4.

[145] David Pointcheval and Olivier Sanders. Short randomizable signatures. pages 111–126, 2016. doi: 10.1007/978-3-319-29485-8_7.

[146] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. 13(3):361–396, June 2000. doi: 10.1007/s001450010003.

[147] Leonid Reyzin, Adam Smith, and Sophia Yakoubov. Turning HATE into LOVE: Homomorphic ad hoc threshold encryption for scalable MPC. Cryptology ePrint Archive, Report 2018/997, 2018. `https://eprint.iacr.org/2018/997`.

[148] Antoine Rondelet and Michal Zajac. Zeth: On integrating zerocash on ethereum. https://arxiv.org/abs/1904.00905, 2019.

[149] Lior Rotem and Gil Segev. Tighter security for schnorr identification and signatures: A high-moment forking lemma for Σ-protocols. pages 222–250, 2021. doi: 10.1007/978-3-030-84242-0_9.

[150] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. pages 543–553, 1999. doi: 10.1109/SFFCS.1999.814628.

[151] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. pages 239–252, 1990. doi: 10.1007/0-387-34805-0_22.

[152] Dominique Schröder and Dominique Unruh. Security of blind signatures revisited. pages 662–679, 2012. doi: 10.1007/978-3-642-30057-8_39.

[153] Nigel P. Smart and Younes Talibi Alaoui. Distributing any elliptic curve based protocol. pages 342–366, 2019. doi: 10.1007/978-3-030-35199-1_17.

[154] Isamu Teranishi and Kazue Sako. k-times anonymous authentication with a constant proving cost. pages 525–542, 2006. doi: 10.1007/11745853_34.

[155] Isamu Teranishi, Jun Furukawa, and Kazue Sako. k-Times anonymous authentication (extended abstract). pages 308–322, 2004. doi: 10.1007/978-3-540-30539-2_22.

[156] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). pages 162–167, 1986. doi: 10.1109/SFCS.1986.25.

[157] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. pages 220–250, 2015. doi: 10.1007/978-3-662-46803-6_8.

[158] Zcash. Zchas documentation. `https://zcash.readthedocs.io`, 2020.