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

Chaya Ganesh[1], Hamidreza Khoshakhlagh[2], Markulf Kohlweiss[3],
Anca Nitulescu[4], and Michał Zając[5]

[1] Indian Institute of Science chaya@iisc.ac.in
[2] Aarhus University hamidreza@cs.au.dk
[3] University of Edinburgh and IOHK mkohlwei@inf.ed.ac.uk
[4] Protocol Labs anca@protocol.ai
[5] Nethermind m.p.zajac@gmail.com

Abstract. We show that three popular universal zero-knowledge SNARKs (Plonk, Sonic, and Marlin) are updatable SRS simulation extractable NIZKs and signatures of knowledge (SoK) out-of-the-box avoiding any compilation overhead.

Towards this we generalize results for the Fiat–Shamir (FS) transformation, which turns interactive protocols into signature schemes, non-interactive proof systems, or SoK in the random oracle model (ROM). The security of the transformation relies on rewinding to extract the secret key or the witness, even in the presence of signing queries for signatures and simulation queries for proof systems and SoK, respectively. We build on this line of work and analyze multi-round FS for arguments with a structured reference string (SRS). The combination of ROM and SRS, while redundant in theory, is the model of choice for the most efficient practical systems to date. We also consider the case where the SRS is updatable and define a strong simulation extractability notion that allows for simulated proofs with respect to an SRS to which the adversary can contribute updates.

We define three properties (trapdoor-less zero-knowledge, rewinding-based knowledge soundness, and a unique response property) that are sufficient for argument systems based on multi-round FS to be also simulation extractable in this strong sense. We show that Plonk, Sonic, and Marlin satisfy these properties, and conjecture that many other argument systems such as Lunar, Basilisk, and transparent variants of Plonk fall within the reach of our main theorem.

1 Introduction

Zero-knowledge proof systems, which allow a prover to convince a verifier of an NP statement $\mathbf{R}(x, w)$ without revealing anything else about the witness w have broad application in cryptography and theory of computation [9, 28, 34]. When restricted to computationally sound proof systems, also called *argument systems*[6], proof size can be shorter than the size of the witness [18]. 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 [47], 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 [11], 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 [12, 14, 15, 32, 37, 38, 43, 44, 49] 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 [20, 22] (sometimes abbreviated SoK or referred to as SNARKY signatures [6, 40]).

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. [11], require a stronger property – *simulation-extractability* (SE) – that corresponds much more closely to existential unforgeability of signatures.

This correspondence is made precise by SoK, which use 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.,

---

[6] We use both terms interchangeably.

Schnorr, can be read as SoK for a specific hard relation, e.g., DL [25]. 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 [22] 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) [13], 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 [8], Fiat–Shamir should be used with care because there are both counterexamples in theory [35] and real-world attacks in practice when implemented incorrectly [48].

In particular, several schemes such as Sonic [46], Plonk [30], Marlin [23] 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 [24, 32, 37, 38, 43, 44, 49] 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. [39] propose a setting to tackle this challenge which allows parties – provers and verifiers – to *update* the SRS.[7] 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 bound on the size of the statement's circuit. Although inefficient, as the SRS size is quadratic in the size of the circuit, [39] 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 [46]. Subsequently, Gabizon, Williamson, and Ciobotaru designed Plonk [30] which currently is the most efficient updatable universal zkSNARK. Independently, Chiesa et al. [23] 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 (srs, x, π) that convinces the verifier of a statement x without knowing a witness. Now, assume that at a later stage, a party produces a triple (srs', x, π') 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 [2, 42] 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.

---

[7] This can be seen as an efficient player-replaceable [33] multi-party computation.

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 model, *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 [40], can be lifted to SoK.
– *Broad applicability.* The updatable SRS plus ROM model includes both the trusted SRS and the ROM model 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. [41] with commitments from Bulletproof [19] and STARKs [10], respectively.

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 protocol from [26]. 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 $\mathsf{P_{FS}}$, $\mathsf{S_{FS}}$, $\mathsf{M_{FS}}$ respectively.

Rewinding-Based Knowledge Soundness (RBKS). Following [26], 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, $(3n+6)$ for Plonk, $(n+1)$ for Sonic , and $(2n+3)$ 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 $\mathsf{Ext_{ks}}$ 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 [27] (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 $\mathsf{P_{FS}}$, $\mathsf{S_{FS}}$, and $\mathsf{M_{FS}}$.

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.

1.3 Related Work

There are many results on simulation extractability for non-interactive zero-knowledge proofs (NIZKs). First, Groth [36] noticed that a (black-box) SE NIZK is universally-composable (UC) [21]. Then Dodis et al. [25] 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 [40] and a SE zkSNARK for QAP was designed by Lipmaa [45]. Kosba et al. [42] give 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. [2] showed another transformation that obtains non-black-box simulation extractability but also preserves the succinctness of the argument. The zkSNARK of [38] has been shown to be SE by introducing minor modifications to the construction and making stronger assumptions [3, 17]. Recently, [6] showed that the Groth's original proof system from [38] 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 [31] 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.

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., [26, 50, 51].

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 [26] 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.

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

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

if $(\texttt{intent} = \texttt{setup}) :$
$\quad (\mathsf{srs}', \rho') \leftarrow \mathsf{GenSRS}(\mathbf{R})$
$\quad Q_{\mathsf{srs}} \leftarrow Q_{\mathsf{srs}} \cup \{(\mathsf{srs}', \rho')\}$
$\quad \mathbf{return}\ (\mathsf{srs}', \rho')$

if $(\texttt{intent} = \texttt{update}) :$
$\quad b \leftarrow \mathsf{VerifySRS}(\mathsf{srs}_n, \{\rho_j\}_{j=1}^n)$
$\quad \mathbf{if}\ (b = 0) : \mathbf{return}\ \bot$
$\quad (\mathsf{srs}', \rho') \leftarrow \mathsf{UpdSRS}(\mathsf{srs}_n, \{\rho_j\}_{j=1}^n)$
$\quad Q_{\mathsf{srs}} \leftarrow Q_{\mathsf{srs}} \cup \{(\mathsf{srs}', \rho')\}$
$\quad \mathbf{return}\ (\mathsf{srs}', \rho')$

if $(\texttt{intent} = \texttt{final}) :$
$\quad b \leftarrow \mathsf{VerifySRS}(\mathsf{srs}_n, \{\rho_j\}_{j=1}^n)$
$\quad \mathbf{if}\ (b = 0) \vee Q_{\mathsf{srs}}^{(2)} \cap \{\rho_j\}_i = \emptyset :$
$\quad \mathbf{return}\ \bot$
$\quad\quad \mathsf{srs} \leftarrow \mathsf{srs}_n, \mathbf{return}\ \mathsf{srs}$
else return $\bot$

Fig. 1: The oracle defines the notion of updatable SRS setup.

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 $\mathsf{R}(\mathcal{A})$.

2.1 Updatable SRS Setup Ceremonies

The definition of updatable SRS ceremonies of [39] requires the following algorithms.

– $(\mathsf{srs}, \rho) \leftarrow \mathsf{GenSRS}(\mathbf{R})$ is a PPT algorithm that takes a relation $\mathbf{R}$ and outputs a reference string srs, and correctness proof ρ.
– $(\mathsf{srs}', \rho') \leftarrow \mathsf{UpdSRS}(\mathsf{srs}, \{\rho_j\}_{j=1}^n)$ 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.[8]

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. 1. The oracle allows the adversary 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 $\texttt{intent} = \texttt{setup}$ 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 $\mathbf{R}$ 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 $\mathbf{R}$ 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.

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:

---
[8] For instance Plonk and Marlin will use the GenSRS, UpdSRS and $\mathsf{VerifySRS}$ algorithms in Fig. 4.

– 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 $\tilde{\pi}[0..j] = (\mathsf{x}, a_1, c_1, \ldots, a_{j-1}, c_{j-1}, a_j)$.[9]
P output 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. $(2\mu + 1)$-message FS-transformed NIZK proof system with an updatable SRS setup

Notation for $(2\mu + 1)$-message Fiat–Shamir transformed proof systems. Let $\mathsf{SRS} = (\mathsf{GenSRS}, \mathsf{UpdSRS}, \mathsf{VerifySRS})$ be the algorithm of an updatable SRS ceremony. All our definitions and theorems are about non-interactive proof systems $\mathbf{\Psi} = (\mathsf{SRS}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ 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.

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

Fig. 2: Simulation oracles: srs is the finalized SRS, only SimO.P′ allows for simulation of false statements

2.3 Trapdoor-Less Zero-Knowledge (TLZK)

We call a protocol *trapdoor-less zero-knowledge* (TLZK) if there exists a simulator 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. 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. $\mathsf{SimO}.\mathsf{P}(x, w)$ and $\mathsf{SimO}.\mathsf{P}'(x)$ 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 1 (Updatable k-Programmable Trapdoor-Less Zero-Knowledge). *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. We call $\mathbf{\Psi}_{\mathsf{FS}}$ 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 $\mathbf{\Psi}_{\mathsf{FS}}$ is trapdoor-less zero-knowledge. Additionally, we say that $\mathbf{\Psi}_{\mathsf{FS}}$ is k-programmable, if Sim before returning a proof π only calls $\mathsf{SimO}.\mathsf{Prog}$ on $(\tilde{\pi}[0..k], h)$. That is, it only programs the k-th message.

Remark 1 (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 $\mathbf{\Psi}$ that is HVZK in the standard model then $\mathbf{\Psi}_{\mathsf{FS}}$ is TLZK. This comes as the simulator Sim in $\mathbf{\Psi}$ produces a valid simulated proof by picking verifier's challenges according to a predefined distribution and $\mathbf{\Psi}_{\mathsf{FS}}$'s simulator $\mathsf{Sim}_{\mathsf{FS}}$ produces its proofs similarly by picking the

---

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

challenges and additionally programming the random oracle to return the picked challenges. Importantly, in both Ψ and Ψ_{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 7, 11 and 14.

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′ [10].

Definition 2 (Updatable Simulation Extractability). *Let* $\Psi_{\mathsf{NI}} = (\mathsf{SRS}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *be a NIZK proof system with an updatable SRS setup. We say that* Ψ_{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 setup oracle* UpdO *and simulation oracle* SimO *and that produces an accepting proof for* Ψ_{NI} *with probability* acc*, where*

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

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

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

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,* $|Q_\mathcal{H}| \le q$, *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 .*

2.5 Unique Response (UR) Protocols

A technical hurdle identified by Faust et al. [26] 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 [26, 27], 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 3 (Updatable k-Unique Response Protocol). *Let* $\Psi_{\mathsf{FS}} = (\mathsf{SRS}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *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* Ψ_{FS} *has* unique responses for k *with security* $\varepsilon_{\mathsf{ur}}(\lambda)$ *if for any* PPT *adversary* $\mathcal{A}_{\mathsf{ur}}$:

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

where srs *is the finalized SRS and* $\mathsf{V}'(\mathsf{srs}, \mathsf{x}, \pi = (a_1, \ldots, a_\mu, a_{\mu+1}))$ *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* Ψ_{FS} *is* k-UR, *if* $\varepsilon_{\mathsf{ur}}(\lambda)$ *is negligible.*

---

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

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,

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 (Tree of accepting transcripts, cf. [16]). *A $(n_1, \ldots, n_\mu)$-tree of accepting transcripts is a tree where each node on depth i, for $i \in [1 \mathinner{..} \mu + 1]$, 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 P, S and 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 5 (Updatable Rewinding-Based Knowledge Soundness). *Let $n_1, \ldots, n_\mu \in \mathbb{N}$. Let $\Psi_{\mathsf{FS}} = (\mathsf{SRS}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ be a $(2\mu + 1)$-message FS-transformed NIZK proof system with an updatable SRS setup for relation $\mathbf{R}$. 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 $\mathsf{Ext}_{\mathsf{ks}}$. Let adversary $\mathcal{A}_{\mathsf{ks}}$ be a PPT algorithm, that outputs a valid proof with probability at least acc, where*

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

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

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

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, $|Q_{\mathcal{H}}| \leq q$.

3 Simulation Extractability—The General Result

Equipped with the definitional framework of Section 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 ensures 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 [4] in later sections.

Theorem 1 (Simulation-extractable multi-message protocols). *Let $\Psi_{\mathsf{FS}} = (\mathsf{SRS}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ be a $(2\mu + 1)$-message FS-transformed NIZK proof system with an updatable SRS setup. If Ψ_{FS} is an updatable k-unique response protocol with security loss $\varepsilon_{\mathsf{ur}}$, is updatable k-programmable trapdoor-less zero-knowledge, and updatable rewinding-based knowledge sound with security loss $\varepsilon_{\mathsf{ks}}$; Then Ψ_{FS} is updatable simulation-extractable with security loss*

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

8

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

Proof. Let $(x, \pi) \leftarrow \mathcal{A}^{\mathsf{UpdO}, \mathsf{SimO}.\mathcal{H}, \mathsf{SimO}.\mathsf{P}'}(r_{\mathcal{A}})$ 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 $\mathbf{R}(x, w)$ 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 $\mathsf{Ext}_{\mathsf{ks}}$ 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. 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. $\tilde{\pi}[i].ch = \mathcal{H}(\tilde{\pi}[0..i])$, for $i < k$.

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

Fig. 3: Simulating random oracle calls.

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 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 1. *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 $\mathbf{\Psi}_{\mathsf{FS}}$.

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 $\tilde{\pi}_{\mathsf{Sim}}[k].ch$ on $\tilde{\pi}_{\mathsf{Sim}}[0..k]$. $\mathcal{A}_{\mathsf{ur}}$ returns two proofs π and π_{Sim} for x: and the challenge $\tilde{\pi}_{\mathsf{Sim}}[k].ch = \tilde{\pi}[k].ch$

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 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 $\mathbf{\Psi}_{\mathsf{FS}}$. This happens only with probability $\varepsilon_{\mathsf{ur}}(\lambda)$. $\qquad\square$

We denote by $\widetilde{\mathsf{acc}}$ the probability that $\mathcal{A}_{\mathsf{ks}}$ outputs an 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 an proof. Thus, the probability that $\mathcal{A}_{\mathsf{ks}}$ outputs an proof is at least $\widetilde{\mathsf{acc}} \geq \mathsf{acc} - \varepsilon_{\mathsf{ur}}(\lambda)$. Since $\mathbf{\Psi}_{\mathsf{FS}}$ is $\varepsilon_{\mathsf{ks}}(\lambda, \widetilde{\mathsf{acc}}, q)$ rewinding-based knowledge sound, there is a tree builder $\mathcal{T}$ and

extractor $\mathsf{Ext_{ks}}$ 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 $\mathsf{Ext_{se}}$ outputs the witness with the same probability.

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

Remark 2. 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 3. 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 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. 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

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

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 $\mathsf{O}_\mathcal{C}$ 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.

$$\begin{array}{ll}
\underline{\mathsf{GenSRS}(1^\lambda, \mathsf{max})} & \underline{\mathsf{UpdSRS}(\mathsf{srs}, \{\rho_j\}_{j=1}^n)} \\[4pt]
\chi \leftarrow\!\!\$ \, \mathbb{F}_p & \text{Parse srs as } \left([\{A_i\}_{i=0}^{\mathsf{max}}]_1, [B]_2\right) \\
\mathsf{srs} := \left(\left[\{\chi^i\}_{i=0}^{\mathsf{max}}\right]_1, [\chi]_2\right); & \chi' \leftarrow\!\!\$ \, \mathbb{F}_p \\
\rho = \left([\chi, \chi]_1, [\chi]_2\right) & \mathsf{srs}' := \left(\left[\{\chi'^i A_i\}_{i=0}^{\mathsf{max}}\right]_1, [\chi' B]_2\right); \\
\mathbf{return} \ (\mathsf{srs}, \rho) & \rho' = \left([\chi' A_1, \chi']_1, [\chi']_2\right) \\
& \mathbf{return} \ (\mathsf{srs}', \rho')
\end{array}$$

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

Parse srs as $\left([\{A_i\}_{i=0}^{\mathsf{max}}]_1, [B]_2\right)$

and $\{\rho_j\}_{j=1}^n$ as $\left\{\left(P_j, \bar{P}_j, \widehat{P}_j\right)\right\}_{j=1}^n$

Verify Update proofs:

$\quad \bar{P}_1 = P_1$

$\quad P_j \bullet [1]_2 = P_{j-1} \bullet \widehat{P}_j \quad \forall j \geq 2$

$\quad \bar{P}_n \bullet [1]_2 = [1]_1 \bullet \widehat{P}_n$

Verify SRS structure:

$\quad [A_i]_1 \bullet [1]_2 = [A_{i-1}]_1 \bullet [B]_2$ for all $0 < i \leq \mathsf{max}$

Fig. 4: Updatable SRS scheme SRS for **PC$_P$**

4.1 Unique Opening Property of 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.

Lemma 2. *Let* **PC$_P$** $=$ (GenSRS, Com, Op, VerifyBatched) *be a batched version of a KZG polynomial commitment, cf. Fig. 5, that commits to m polynomials of degree up to* max. *Let* $\mathbf{z} = (z_0, \ldots, z_{l-1}) \in \mathbb{F}_p^l$ *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* $\mathbf{s_{K_i}} \in \mathbb{F}_p^{k_i}$ *the evaluations of polynomials at* z_i, *and* $\mathbf{o} = (o_0, \ldots, o_{l-1}) \in \mathbb{F}_p^l$ *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* $\varepsilon_{\mathsf{op}}(\lambda) \leq l \cdot \varepsilon_{\mathsf{udlog}}(\lambda) + 1/p$, *where* $\varepsilon_{\mathsf{udlog}}(\lambda)$ *is security of the* $(\mathsf{max}, 1)$-udlog *assumption and p is the field used in* **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 $\left([1, \ldots, \chi'^{\mathsf{max}}]_1, [1]_2\right)$. We consider verification equation (**) as a polynomial in X and the verification equation (*) as it's evaluation at χ'. Consider an opening such that verification equation (**), cf. Fig. 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 $\varepsilon_{\mathsf{udlog}}(\lambda)$.

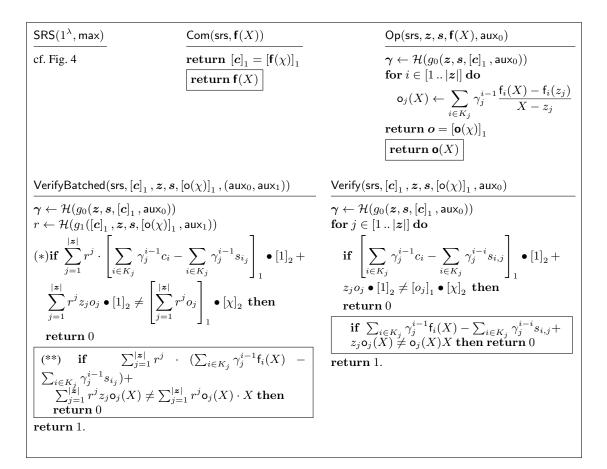SRS(1^λ, max)

cf. Fig. 4

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

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

$\boxed{\mathbf{return}\ \mathbf{f}(X)}$

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

$\boldsymbol{\gamma} \leftarrow \mathcal{H}(g_0(\boldsymbol{z}, \boldsymbol{s}, [\boldsymbol{c}]_1, \mathsf{aux}_0))$
$\mathbf{for}\ i \in [1 .. |\boldsymbol{z}|]\ \mathbf{do}$

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

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

$\boxed{\mathbf{return}\ \mathbf{o}(X)}$

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

$\boldsymbol{\gamma} \leftarrow \mathcal{H}(g_0(\boldsymbol{z}, \boldsymbol{s}, [\boldsymbol{c}]_1, \mathsf{aux}_0))$
$r \leftarrow \mathcal{H}(g_1([\boldsymbol{c}]_1, \boldsymbol{z}, \boldsymbol{s}, [\mathsf{o}(\chi)]_1, \mathsf{aux}_1))$

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

$\displaystyle\sum_{j=1}^{|\boldsymbol{z}|} r^j z_j o_j \bullet [1]_2 \neq \left[\sum_{j=1}^{|\boldsymbol{z}|} r^j o_j \right]_1 \bullet [\chi]_2\ \mathbf{then}$

   $\mathbf{return}\ 0$

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

$\mathbf{return}\ 1.$

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

$\boldsymbol{\gamma} \leftarrow \mathcal{H}(g_0(\boldsymbol{z}, \boldsymbol{s}, [\boldsymbol{c}]_1, \mathsf{aux}_0))$
$\mathbf{for}\ j \in [1 .. |\boldsymbol{z}|]\ \mathbf{do}$

   $\mathbf{if}\ \left[\displaystyle\sum_{i \in K_j} \gamma_j^{i-1} c_i - \sum_{i \in K_j} \gamma_j^{i-i} s_{i,j} \right]_1 \bullet [1]_2 +$

   $z_j o_j \bullet [1]_2 \neq [o_j]_1 \bullet [\chi]_2\ \mathbf{then}$

      $\mathbf{return}\ 0$

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

$\mathbf{return}\ 1.$

Fig. 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$'$.

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:

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

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

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

is q/p cf. [30]. When $r^j \left(\sum_{i \in K_j} \gamma^{i-1} \cdot \mathsf{f}_i(X) - \sum_{i \in K_j} \gamma^{i-1} \cdot s_{ij} \right) = r^j \mathsf{o}_j(X)(X - z_j)$ holds, polynomial $\mathsf{o}_j(X)$ 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 $\varepsilon_{\mathsf{udlog}}(\lambda) + \frac{q}{p}$. □

5   Concrete SNARKs Preliminaries

Bilinear groups. A bilinear group generator $\mathsf{Pgen}(1^\lambda)$ returns public parameters $\mathsf{p} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2)$, where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are additive cyclic groups of prime order $p = 2^{\Omega(\lambda)}$, $[1]_1, [1]_2$ 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. We assume the bilinear pairing to be Type-3, i.e., that there is no efficient isomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$ or from $\mathbb{G}_2$ to $\mathbb{G}_1$. We use the by now standard bracket notation, i.e., we write $[a]_\iota$ to denote $a[1]_\iota$. We denote $\hat{e}([a]_1, [b]_2)$ as $[a]_1 \bullet [b]_2$. Thus, $[a]_1 \bullet [b]_2 = [ab]_T$. 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 Pgen.

5.1   Algebraic Group Model

The algebraic group model (AGM) of Fuchsbauer, Kiltz, and Loss [29] 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 $[\mathsf{p}_1(\chi), \ldots, \mathsf{p}_\mathsf{k}(\chi)]_1$, for some polynomials $\mathsf{p}_1(X), \ldots, \mathsf{p}_\mathsf{k}(X) \in \mathbb{F}_p[X]$. 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 [30] 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.

5.2   Dlog Assumptions in Standard and Updatable Setting

Definition 6 ((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. 1. The honestly generated SRS is set to be a dlog challenge and the update algorithm UpdSRS 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.

Definition 7 ((q_1, q_2)-udlog assumption). *Let $\mathcal{A}$ be a* PPT *adversary that gets oracle access to* UpdO *with internal algorithms* (GenSRS, UpdSRS, VerifySRS)*, where* GenSRS *and* UpdSRS *are defined as follows:*

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

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

Remark 4 (Single adversarial updates after an honest setup.). As an alternative to the updatable setting defined in Fig. 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 [39] 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 3. (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. $\square$

Generalized Forking Lemma Although dubbed "general", the forking lemma of [7] 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 **P** we need at least $(3n + 6)$ valid proofs (where n is the number of constrains), $(n + 1)$ for **S**, and $2n + 3$ for **M**. Here we use a result by Attema et al. [4][11] which lower-bounds the probability of generating a tree of accepting transcripts T. We restate their Proposition 2 in our notation:

Lemma 4 (Run Time and Success Probability). *Let $N = n_1 \cdots n_\mu$ 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 $(n_1, \ldots, n_\mu)$-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 8 (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*

---

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

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 4.1.

Remark 5. Fig. 5, cf. Section 4.1, presents efficient variants of KZG [41] 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.

6 Non-malleability of Plonk

In this section, we show that $\mathsf{P_{FS}}$ is simulation-extractable. To this end, we first use the unique opening property to show that $\mathsf{P_{FS}}$ has the 3-UR property, cf. Lemma 5. Next, we show that $\mathsf{P_{FS}}$ 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 6. We then show that $\mathsf{P_{FS}}$ is 3-programmable trapdoor-less ZK in the AGM, cf. Lemma 7.

Given rewinding-based knowledge soundness, 3-UR and trapdoor-less zero-knowledge of $\mathsf{P_{FS}}$, we invoke Theorem 1 and conclude that $\mathsf{P_{FS}}$ is simulation-extractable.

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.[12]

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 $(\mathsf{x}_j)_{j=1}^{\ell}$ 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.

Plonk *SRS generating algorithm* $\mathsf{GenSRS}(\mathbf{R})$*:* 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 $\mathsf{S}_{\mathsf{id}j}, \mathsf{S}_{\sigma j}$, for $\mathsf{j} \in [1..3]$, are computed:

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

---

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

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 $3 \cdot n$ different elements. Gabizon et al. [30] 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 q_L, q_R, q_O, q_M, q_C such that

$$
\begin{aligned}
q_L(X) &= \sum_{i=1}^{n} \boldsymbol{q_{Li}} L_i(X), & q_O(X) &= \sum_{i=1}^{n} \boldsymbol{q_{Oi}} L_i(X), \\
q_R(X) &= \sum_{i=1}^{n} \boldsymbol{q_{Ri}} L_i(X), & q_C(X) &= \sum_{i=1}^{n} \boldsymbol{q_{Ci}} L_i(X). \\
q_M(X) &= \sum_{i=1}^{n} \boldsymbol{q_{Mi}} L_i(X),
\end{aligned}
$$

Proving statements in P_{FS} We show how prover's algorithm $\mathsf{P}(\mathsf{srs}, \mathsf{x} = (\mathsf{w}'_i)_{i=1}^{\ell}, \mathsf{w} = (\mathsf{w}_i)_{i=1}^{3 \cdot n})$ 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 $a(X), b(X), c(X)$ as

$$
\begin{aligned}
a(X) &= (b_1 X + b_2) Z_H(X) + \sum_{i=1}^{n} \mathsf{w}_i L_i(X) \\
b(X) &= (b_3 X + b_4) Z_H(X) + \sum_{i=1}^{n} \mathsf{w}_{n+i} L_i(X) \\
c(X) &= (b_5 X + b_6) Z_H(X) + \sum_{i=1}^{n} \mathsf{w}_{2 \cdot n+i} L_i(X)
\end{aligned}
$$

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

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 $z(X)$

$$
\begin{aligned}
z(X) = {} & (b_7 X^2 + b_8 X + b_9) Z_H(X) + L_1(X) + \\
& + \sum_{i=1}^{n-1} \left(L_{i+1}(X) \prod_{j=1}^{i} \frac{(\mathsf{w}_j + \beta \omega^{j-1} + \gamma)(\mathsf{w}_{n+j} + \beta k_1 \omega^{j-1} + \gamma)(\mathsf{w}_{2n+j} + \beta k_2 \omega^{j-1} + \gamma)}{(\mathsf{w}_j + \sigma(j)\beta + \gamma)(\mathsf{w}_{n+j} + \sigma(n+j)\beta + \gamma)(\mathsf{w}_{2n+j} + \sigma(2n+j)\beta + \gamma)} \right)
\end{aligned}
$$

Output polynomial commitment $[z(\chi)]_1$.

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

$$
\begin{aligned}
t(X) = {} & \\
& (a(X)b(X)q_M(X) + a(X)q_L(X) + b(X)q_R(X) + c(X)q_O(X) + PI(X) + q_C(X))/Z_H(X) + \\
& + ((a(X) + \beta X + \gamma)(b(X) + \beta k_1 X + \gamma)(c(X) + \beta k_2 X + \gamma)z(X))\alpha/Z_H(X) \\
& - (a(X) + \beta S_{\sigma 1}(X) + \gamma)(b(X) + \beta S_{\sigma 2}(X) + \gamma)(c(X) + \beta S_{\sigma 3}(X) + \gamma)z(X\omega))\alpha/Z_H(X) \\
& + (z(X) - 1)L_1(X)\alpha^2/Z_H(X)
\end{aligned}
$$

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

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

$$
\begin{aligned}
r(X) = {} & a(\mathfrak{z})b(\mathfrak{z})q_M(X) + a(\mathfrak{z})q_L(X) + b(\mathfrak{z})q_R(X) + c(\mathfrak{z})q_O(X) + q_C(X) \\
& + \alpha \cdot ((a(\mathfrak{z}) + \beta\mathfrak{z} + \gamma)(b(\mathfrak{z}) + \beta k_1\mathfrak{z} + \gamma)(c(\mathfrak{z}) + \beta k_2\mathfrak{z} + \gamma) \cdot z(X)) \\
& - \alpha \cdot ((a(\mathfrak{z}) + \beta S_{\sigma 1}(\mathfrak{z}) + \gamma)(b(\mathfrak{z}) + \beta S_{\sigma 2}(\mathfrak{z}) + \gamma)\beta z(\mathfrak{z}\omega) \cdot S_{\sigma 3}(X)) \\
& + \alpha^2 \cdot L_1(\mathfrak{z}) \cdot z(X)
\end{aligned}
$$

Output $a(\mathfrak{z}), b(\mathfrak{z}), c(\mathfrak{z}), S_{\sigma 1}(\mathfrak{z}), S_{\sigma 2}(\mathfrak{z}), t(\mathfrak{z}), z(\mathfrak{z}\omega), r(\mathfrak{z})$.

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

$$
\begin{aligned}
W_{\mathfrak{z}}(X) &= \frac{1}{X - \mathfrak{z}} \left(\begin{aligned} & t_{lo}(X) + \mathfrak{z}^n t_{mid}(X) + \mathfrak{z}^{2n} t_{hi}(X) - t(\mathfrak{z}) + v(r(X) - r(\mathfrak{z})) + v^2(a(X) - a(\mathfrak{z})) \\ & + v^3(b(X) - b(\mathfrak{z})) + v^4(c(X) - c(\mathfrak{z})) + v^5(S_{\sigma 1}(X) - S_{\sigma 1}(\mathfrak{z})) \\ & + v^6(S_{\sigma 2}(X) - S_{\sigma 2}(\mathfrak{z})) \end{aligned} \right) \\
W_{\mathfrak{z}\omega}(X) &= (z(X) - z(\mathfrak{z}\omega))/(X - \mathfrak{z}\omega)
\end{aligned}
$$

Output $[W_{\mathfrak{z}}(\chi), W_{\mathfrak{z}\omega}(\chi)]_1$.

Plonk verifier $V(srs, x, \pi)$**:**

The Plonk verifier works as follows

1. Validate all obtained group elements.
2. Validate all obtained field elements.
3. Parse the instance as $\{w_i\}_{i=1}^{\ell} \leftarrow x$.
4. Compute challenges $\beta, \gamma, \alpha, \mathfrak{z}, v, u$ from the transcript.
5. Compute zero polynomial evaluation $Z_H(\mathfrak{z}) = \mathfrak{z}^n - 1$.
6. Compute Lagrange polynomial evaluation $L_1(\mathfrak{z}) = \frac{\mathfrak{z}^n - 1}{n(\mathfrak{z}-1)}$.
7. Compute public input polynomial evaluation $PI(\mathfrak{z}) = \sum_{i \in [1 .. \ell]} w_i L_i(\mathfrak{z})$.
8. Compute quotient polynomials evaluations

$$t(\mathfrak{z}) = \Big(r(\mathfrak{z}) + PI(\mathfrak{z}) - (a(\mathfrak{z}) + \beta S_{\sigma 1}(\mathfrak{z}) + \gamma)(b(\mathfrak{z}) + \beta S_{\sigma 2}(\mathfrak{z}) + \gamma)(c(\mathfrak{z}) + \gamma)z(\mathfrak{z}\omega)\alpha - L_1(\mathfrak{z})\alpha^2\Big)/Z_H(\mathfrak{z}).$$

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

$$[D]_1 = v \begin{pmatrix} a(\mathfrak{z})b(\mathfrak{z}) \cdot [q_M]_1 + a(\mathfrak{z})\,[q_L]_1 + b\,[q_R]_1 + c\,[q_O]_1 + \\ + ((a(\mathfrak{z}) + \beta\mathfrak{z} + \gamma)(b(\mathfrak{z}) + \beta k_1 \mathfrak{z} + \gamma)(c + \beta k_2 \mathfrak{z} + \gamma)\alpha + L_1(\mathfrak{z})\alpha^2) + \\ - (a(\mathfrak{z}) + \beta S_{\sigma 1}(\mathfrak{z}) + \gamma)(b(\mathfrak{z}) + \beta S_{\sigma 2}(\mathfrak{z}) + \gamma)\alpha\beta z(\mathfrak{z}\omega)\,[S_{\sigma 3}(\chi)]_1) \end{pmatrix} + $$
$$ + u\,[z(\chi)]_1.$$

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

$$[F]_1 = \left([t_{lo}(\chi)]_1 + \mathfrak{z}^n\,[t_{mid}(\chi)]_1 + \mathfrak{z}^{2n}\,[t_{hi}(\chi)]_1\right) + u\,[z(\chi)]_1 + $$
$$ + v \begin{pmatrix} a(\mathfrak{z})b(\mathfrak{z}) \cdot [q_M]_1 + a(\mathfrak{z})\,[q_L]_1 + b(\mathfrak{z})\,[q_R]_1 + c(\mathfrak{z})\,[q_O]_1 + \\ + ((a(\mathfrak{z}) + \beta\mathfrak{z} + \gamma)(b(\mathfrak{z}) + \beta k_1 \mathfrak{z} + \gamma)(c(\mathfrak{z}) + \beta k_2 \mathfrak{z} + \gamma)\alpha + L_1(\mathfrak{z})\alpha^2) + \\ - (a(\mathfrak{z}) + \beta S_{\sigma 1}(\mathfrak{z}) + \gamma)(b(\mathfrak{z}) + \beta S_{\sigma 2}(\mathfrak{z}) + \gamma)\alpha\beta z(\mathfrak{z}\omega)\,[S_{\sigma 3}(\chi)]_1) \end{pmatrix}$$
$$ + v^2\,[a(\chi)]_1 + v^3\,[b(\chi)]_1 + v^4\,[c(\chi)]_1 + v^5\,[S_{\sigma 1}(\chi)]_1 + v^6\,[S_{\sigma 2}(\chi)]_1.$$

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

$$[E]_1 = \frac{1}{Z_H(\mathfrak{z})} \begin{bmatrix} r(\mathfrak{z}) + PI(\mathfrak{z}) + \alpha^2 L_1(\mathfrak{z}) + \\ - \alpha\left((a(\mathfrak{z}) + \beta S_{\sigma 1}(\mathfrak{z}) + \gamma)(b(\mathfrak{z}) + \beta S_{\sigma 2}(\mathfrak{z}) + \gamma)(c(\mathfrak{z}) + \gamma)z(\mathfrak{z}\omega)\right) \end{bmatrix}_1$$
$$ + \left[vr(\mathfrak{z}) + v^2 a(\mathfrak{z}) + v^3 b(\mathfrak{z}) + v^4 c(\mathfrak{z}) + v^5 S_{\sigma 1}(\mathfrak{z}) + v^6 S_{\sigma 2}(\mathfrak{z}) + uz(\mathfrak{z}\omega)\right]_1.$$

12. Check whether the verification equation holds

$$\left([W_{\mathfrak{z}}(\chi)]_1 + u \cdot [W_{\mathfrak{z}\omega}(\chi)]_1\right) \bullet [\chi]_2 - $$
$$\left(\mathfrak{z} \cdot [W_{\mathfrak{z}}(\chi)]_1 + u\mathfrak{z}\omega \cdot [W_{\mathfrak{z}\omega}(\chi)]_1 + [F]_1 - [E]_1\right) \bullet [1]_2 = 0. \quad (2)$$

The verification equation is a batched version of the verification equation from [41] 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 $\mathsf{Sim}_\chi(srs, td = \chi, x)$**:** We describe the simulator in Lemma 7.

6.2 Simulation extractability of Plonk

Due to lack of space, we provide here only theorem statements and intuition for why they hold. Full proofs are given in **??**.

Unique Response Property

Lemma 5. *Let* $\mathsf{PC_P}$ *be a polynomial commitment that is* $\varepsilon_{\mathsf{bind}}(\lambda)$*-binding and has unique opening property with loss* $\varepsilon_{\mathsf{op}}(\lambda)$*. Then* $\mathsf{P_{FS}}$ *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 $\mathfrak{z}$, 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 $S_{\sigma1}(\mathfrak{z}), S_{\sigma2}(\mathfrak{z})$, and evaluation of a linearization polynomial $\mathsf{r}(\mathfrak{z})$.

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 [30], $\varepsilon_{\mathsf{op}}(\lambda) \leq \varepsilon_{\mathsf{udlog}}(\lambda) + q/p$, cf. Lemma 2.)

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 6. $\mathbf{P}_{\mathsf{FS}}$ *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. [5, 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 that outputs a tree T, and let $\mathsf{Ext}_{\mathsf{ks}}$ 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 $\mathfrak{z}$ is presented, thus tree T is built using different values of $\mathfrak{z}$. 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 $\mathsf{Ext}_{\mathsf{ks}}$ does not manage to output a witness w such that $\mathbf{R}(\mathsf{x}, \mathsf{w})$ 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 probability that Game 1 is aborted, while Game 0 is not, is at most

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

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 $(n+5,1)$-udlog instance $\left[1, \ldots, \chi^{n+5}\right]_1, [\chi]_2$. Then it can update the instance to another one $\left[1, \ldots, \chi'^{n+5}\right]_1, [\chi']_2$. 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 $\left[1, \ldots, \chi'^{n+5}\right]_1, [\chi']_2$. 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 $(3n+6) \cdot \varepsilon_{\mathsf{udlog}}(\lambda)$.

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 $(3n+6)$ accepting transcripts for different challenges $\mathfrak{z}$, 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. $\mathsf{Ext}_{\mathsf{ks}}$ can recreate polynomials' coefficients by interpolation and reveal the witness given $(3n+6)$ evaluations.

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

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

$\square$

Trapdoor-Less Zero-Knowledge of Plonk

Lemma 7. 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 $\mathfrak{z}$ and a polynomial $\tilde{\mathsf{t}}(X)$ such that $\mathsf{t}(\mathfrak{z}) = \tilde{\mathsf{t}}(\mathfrak{z})$. 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 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 $w_i = 0$ for $i \in [1 .. 3n]$. Then Sim outputs $[a(\chi), b(\chi), c(\chi)]_1$. For the first challenge, the simulator picks permutation argument challenges β, γ randomly.

For its 2-nd message, the simulator computes $z(X)$ from the newly picked randomizers b_7, b_8, b_9 and coefficients of polynomials $a(X), b(X), c(X)$. Then it evaluates $z(X)$ honestly and outputs $[z(\chi)]_1$. 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 $\mathfrak{z}$, 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 $a(\mathfrak{z}), b(\mathfrak{z}), c(\mathfrak{z}), S_{\sigma 1}(\mathfrak{z}), S_{\sigma 2}(\mathfrak{z}), PI(\mathfrak{z}), L_1(\mathfrak{z}), Z_H(\mathfrak{z}), z(\mathfrak{z}\omega)$ and computes $t(X)$ honestly. Since for a random $a(X), b(X), c(X), z(X)$ the constraint system is (with overwhelming probability) not satisfied and the constraints-related polynomials are not divisible by $Z_H(X)$, hence $t(X)$ is a rational function rather than a polynomial. Then, the simulator evaluates $t(X)$ at $\mathfrak{z}$ and picks randomly a degree-$(3n + 15)$ polynomial $\tilde{t}(X)$ such that $t(\mathfrak{z}) = \tilde{t}(\mathfrak{z})$ and publishes a commitment $[\tilde{t}_{lo}(\chi), \tilde{t}_{mid}(\chi), \tilde{t}_{hi}(\chi)]_1$. After that the simulator outputs $\mathfrak{z}$ as a challenge.

For the next message, the simulator computes polynomial $r(X)$ as an honest prover would, cf. Section 6.1 and evaluates $r(X)$ at $\mathfrak{z}$.

The rest of the evaluations are already computed, thus Sim simply outputs $a(\mathfrak{z}), b(\mathfrak{z}), c(\mathfrak{z}), S_{\sigma 1}(\mathfrak{z}), S_{\sigma 2}(\mathfrak{z}), t(\mathfrak{z}), z(\mathfrak{z}\omega)$. 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. $\square$

Simulation Extractability of P_{FS}

Since Lemmas 5 to 7 hold, **P** is 3-UR, rewinding-based knowledge sound and trapdoor-less zero-knowledge. We now make use of Theorem 1 and show that P_{FS} is simulation-extractable as defined in Definition 2.

Corollary 1 (Simulation extractability of P_{FS}). P_{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*

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

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

7 Non-malleability of S_{FS}

7.1 Preliminaries

Definition 9 ((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*

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

Definition 10 ((q_1, q_2)-uldlog assumption). *Let $\mathcal{A}$ be a* PPT *adversary that gets oracle access to* UpdO *with internal algorithms* (GenSRS$_{ldlog}$, UpdSRS$_{ldlog}$, VerifySRS)*, where* GenSRS$_{ldlog}$ *and* UpdSRS$_{ldlog}$ *are defined as follows:*

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

Then

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

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

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.

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

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

return $\left[\{\chi^i\}_{i=-\mathsf{Q_{mult}}}^{\mathsf{Q_{mult}}}, \{\alpha\chi^i\}_{i=-\mathsf{Q_{mult}}, i\neq0}^{\mathsf{Q_{mult}}}\right]_1,$

$\qquad \left[\{\chi^i, \alpha\chi^i\}_{i=-\mathsf{Q_{mult}}}^{\mathsf{Q_{mult}}}\right]_2, [\alpha]_T$

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

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

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

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

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

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

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

if $[\mathsf{o}(\chi)]_1 \bullet [\alpha\chi]_2 + [s - z\mathsf{o}(\chi)]_1 \bullet [\alpha]_2 =$

$\quad [c]_1 \bullet \left[\chi^{\mathsf{-d+max}}\right]_2$ **then return** 1

else return 0.

Fig. 6: **PC$_\mathsf{S}$** polynomial commitment scheme.

The Constraint System Fig. 6 presents a variant of KZG [41] polynomial commitment schemes used in Sonic. Sonic's system of constraints composes of three $\mathsf{Q_{mult}}$-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 $\mathsf{Q_{lin}}$ 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

$$\mathsf{u_i}(Y) = \sum_{q=1}^{\mathsf{Q_{lin}}} Y^{q+\mathsf{Q_{mult}}} u_{q,i}, \qquad \mathsf{w_i}(Y) = -Y^i - Y^{-i} + \sum_{q=1}^{\mathsf{Q_{lin}}} Y^{q+\mathsf{Q_{mult}}} w_{q,i},$$

$$\mathsf{v_i}(Y) = \sum_{q=1}^{\mathsf{Q_{lin}}} Y^{q+\mathsf{Q_{mult}}} v_{q,i}, \qquad \mathsf{k}(Y) = \sum_{q=1}^{\mathsf{Q_{lin}}} Y^{q+\mathsf{Q_{mult}}} k_q. \tag{3}$$

Sonic constraint system requires that

$$\boldsymbol{a}^\top \cdot \mathbf{u}(Y) + \boldsymbol{b}^\top \cdot \mathbf{v}(Y) + \boldsymbol{c}^\top \cdot \mathbf{w}(Y) + \sum_{i=1}^{\mathsf{Q_{mult}}} a_i b_i (Y^i + Y^{-i}) - \mathsf{k}(Y) = 0. \tag{4}$$

In Sonic we will use commitments to the following polynomials.

$$\mathsf{r}(X, Y) = \sum_{i=1}^{\mathsf{Q_{mult}}} \left(a_i X^i Y^i + b_i X^{-i} Y^{-i} + c_i X^{-i-\mathsf{Q_{mult}}} Y^{-i-\mathsf{Q_{mult}}}\right)$$

$$\mathsf{s}(X, Y) = \sum_{i=1}^{\mathsf{Q_{mult}}} \left(u_i(Y) X^{-i} + v_i(Y) X^i + w_i(Y) X^{i+\mathsf{Q_{mult}}}\right)$$

$$\mathsf{t}(X, Y) = \mathsf{r}(X, 1)(\mathsf{r}(X, Y) + \mathsf{s}(X, Y)) - \mathsf{k}(Y).$$

Polynomials $r(X, Y), s(X, Y), t(X, Y)$ are designed such that $t(0, Y) = \boldsymbol{a}^\top \cdot \mathbf{u}(Y) + \boldsymbol{b}^\top \cdot \mathbf{v}(Y) + \boldsymbol{c}^\top \cdot \mathbf{w}(Y) + \sum_{i=1}^{Q_{\mathsf{mult}}} a_i b_i (Y^i + Y^{-i}) - \mathsf{k}(Y)$. That is, the prover is asked to show that $t(0, Y) = 0$, cf. Eq. (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* GenSRS($\mathbf{R}$). 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 $c_{Q_{\mathsf{mult}}+1}, c_{Q_{\mathsf{mult}}+2}, c_{Q_{\mathsf{mult}}+3}, c_{Q_{\mathsf{mult}}+4} \leftarrow\!\!\$ \, \mathbb{F}_p$. Sets $r(X, Y) \leftarrow r(X, Y) + \sum_{i=1}^4 c_{Q_{\mathsf{mult}}+i} X^{-2Q_{\mathsf{mult}}-i}$. Commits to $r(X, 1)$ and outputs $[r]_1 \leftarrow \mathsf{Com}(\mathsf{srs}, Q_{\mathsf{mult}}, r(X, 1))$. Then it computes challenge $y = \mathcal{H}(\tilde{\pi}[0..1])$.
Message 2 P commits to $t(X, y)$ and outputs $[t]_1 \leftarrow \mathsf{Com}(\mathsf{srs}, \mathsf{d}, t(X, y))$. Then it gets a challenge $z = \mathcal{H}(\tilde{\pi}[0..2])$.
Message 3 The prover computes commitment openings. That is, it outputs

$$[o_a]_1 = \mathsf{Op}(\mathsf{srs}, z, r(z, 1), r(X, 1))$$
$$[o_b]_1 = \mathsf{Op}(\mathsf{srs}, yz, r(yz, 1), r(X, 1))$$
$$[o_t]_1 = \mathsf{Op}(\mathsf{srs}, z, t(z, y), t(X, y))$$

along with evaluations $a' = r(z, 1), b' = r(y, z), t' = t(z, y)$. Then it engages in the signature of correct computation playing the role of the helper, i.e. it commits to $s(X, y)$ and sends the commitment $[s]_1$, commitment opening

$$[o_s]_1 = \mathsf{Op}(\mathsf{srs}, z, s(z, y), s(X, y)),$$

and $s' = s(z, y)$. Then it obtains a challenge $u = \mathcal{H}(\tilde{\pi}[0..3])$.
Message 4 For the next message the prover computes $[c]_1 \leftarrow \mathsf{Com}(\mathsf{srs}, \mathsf{d}, s(u, Y))$ and computes commitments' openings

$$[w]_1 = \mathsf{Op}(\mathsf{srs}, u, s(u, y), s(X, y)),$$
$$[q_y]_1 = \mathsf{Op}(\mathsf{srs}, y, s(u, y), s(u, Y)),$$

and returns $[w]_1, [q_y]_1, s = s(u, y)$. Eventually the prover gets the last challenge $z' = \mathcal{H}(\tilde{\pi}[0..4])$.
Message 5 For the final message, P computes opening $[q_{z'}]_1 = \mathsf{Op}(\mathsf{srs}, z', s(u, z'), s(u, X))$ 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:

$\mathsf{PC_S}.\mathsf{V}(\mathsf{srs}, Q_{\mathsf{mult}}, [r]_1, z, a', [o_a]_1)$,
$\mathsf{PC_S}.\mathsf{V}(\mathsf{srs}, Q_{\mathsf{mult}}, [r]_1, yz, b', [o_b]_1)$,
$\mathsf{PC_S}.\mathsf{V}(\mathsf{srs}, \mathsf{d}, [t]_1, z, t', [o_t]_1)$,
$\mathsf{PC_S}.\mathsf{V}(\mathsf{srs}, \mathsf{d}, [s]_1, z, s', [o_s]_1)$,

$\mathsf{PC_S}.\mathsf{V}(\mathsf{srs}, \mathsf{d}, [s]_1, u, s, [w]_1)$,
$\mathsf{PC_S}.\mathsf{V}(\mathsf{srs}, \mathsf{d}, [c]_1, y, s, [q_y]_1)$,
$\mathsf{PC_S}.\mathsf{V}(\mathsf{srs}, \mathsf{d}, [c]_1, z', s(u, z'), [q_{z'}]_1)$,

and accepts the proof iff all the checks holds. Note that the value $s(u, z')$ 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.

7.3 Unique Opening Property of $\mathsf{PC_S}$

Lemma 8. $\mathsf{PC_S}$ *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

$$\Pr \left[\begin{array}{l} \mathsf{Verify}(\mathsf{srs}, [c]_1, z, s, [o]_1) = 1, \\ \mathsf{Verify}(\mathsf{srs}, [c]_1, z, \tilde{s}, [\tilde{o}]_1) = 1 \end{array} \middle| ([c]_1, z, s, [o]_1, [\tilde{o}]_1) \leftarrow \mathcal{A}^{\mathsf{UpdO}}(1^\lambda, \mathsf{max}) \right]$$

is at most negligible.

As noted in [30, 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

$$\alpha(X^{\mathsf{d-max}} f(X) \cdot X^{-\mathsf{d+max}} - s) \equiv \alpha \cdot o(X)(X - z), \tag{5}$$

what is equivalent to

$$f(X) - s \equiv o(X)(X - z). \tag{6}$$

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. $\square$

7.4 Unique Response Property

The unique response property of S_{FS} follows from the unique opening property of the polynomial commitment scheme $\mathsf{PC_S}$.

Lemma 9. *If a polynomial commitment scheme $\mathsf{PC_S}$ is evaluation binding with security loss $\varepsilon_{\mathsf{bind}}(\lambda)$ and has unique openings property with security loss $\varepsilon_{\mathsf{op}}(\lambda)$, then 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 S_{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 $2\varepsilon_{\mathsf{bind}}(\lambda) + \varepsilon_{\mathsf{op}}(\lambda)$. $\square$

7.5 Rewinding-Based Knowledge Soundness

Lemma 10. S_{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* $(\mathsf{max}, \mathsf{max})$*-uldlog assumption.*

Let $\mathcal{A}^{\mathcal{H}, \mathsf{UpdO}}(1^\lambda; r)$ be the adversary who outputs (x, π) such that $\mathsf{S}_{\mathsf{FS}}.\mathsf{V}$ accepts the proof. Let $\mathcal{T}$ be the tree-building algorithm of Lemma 4 that outputs a tree T, and let $\mathsf{Ext}_{\mathsf{ks}}$ 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 $\mathsf{Ext}_{\mathsf{ks}}$ does not manage to output a witness w such that $\mathbf{R}(\mathsf{x}, \mathsf{w})$ 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 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 $[\chi^{-\mathsf{max}}, \ldots, 1, \ldots, \chi^{\mathsf{max}}]_1, [\chi^{-\mathsf{max}}, \ldots, 1, \ldots, \chi^{\mathsf{max}}]_2$. Then it can update the instance to another one $[\chi'^{-\mathsf{max}}, \ldots, 1, \ldots, \chi'^{\mathsf{max}}]_1, [\chi'^{-\mathsf{max}}, \ldots, 1, \ldots, \chi'^{\mathsf{max}}]_2$. 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 $[\chi'^{-\mathsf{max}}, \ldots, 1, \ldots, \chi'^{\mathsf{max}}]_1, [\chi'^{-\mathsf{max}}, \ldots, 1, \ldots, \chi'^{\mathsf{max}}]_2$. 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. [46].

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

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

7.6 Trapdoor-Less Zero-Knowledge of Sonic

Lemma 11. 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

$$\boldsymbol{c} = \boldsymbol{a} \cdot \boldsymbol{b}. \tag{7}$$

2. Pick randomizers $c_{Q_{mult}+1}, \ldots, c_{Q_{mult}+4}$, honestly compute polynomials $r(X, Y), r'(X, Y), s(X, Y)$ and pick randomly challenges y, z.
3. Output commitment $[r]_1 \leftarrow \mathsf{Com}(\mathsf{srs}, \mathsf{Q}_{mult}, \mathsf{r}(X, 1))$ 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). $\square$

Remark 6. As noted in [46], Sonic is statistically subversion zero-knowledge (Sub-ZK). As noted in [1], 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 11, 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.

7.7 Simulation Extractability of S_{FS}

Since Lemmas 9 to 11 hold, S_{FS} is 2-UR, rewinding-based knowledge sound and trapdoor-less zero-knowledge. We now make use of Theorem 1 and show that S_{FS} is simulation-extractable as defined in Definition 2.

Corollary 2 (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*

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

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

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.

8.1 Marlin **Protocol Rolled-out**

Marlin uses R1CS as its arithmetization method. Given instance x, witness w and $|H| \times |H|$ 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 $|K|$ non-zero entries. Obviously, $|K| \le |H|^2$. 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 $\{|H| + 2b - 1, 2|H| + b - 1, 6|K| - 6\}$.

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 $\mathsf{z}_A(X), \mathsf{z}_B(X), \mathsf{z}_C(X)$ which encode vectors $\boldsymbol{Az}, \boldsymbol{Bz}, \boldsymbol{Cz}$ and have degree $< |H|$. Importantly, when constraints are fulfilled, $\mathsf{z}_A(X)\mathsf{z}_B(X) - \mathsf{z}_C(X) = \mathsf{h}_0(X)\mathsf{Z}_H(X)$, for some $\mathsf{h}_0(X)$ and vanishing polynomial $\mathsf{Z}_H(X)$. The prover sends commitments to these polynomials and shows that they have been computed correctly. More precisely, it shows that

$$\forall \boldsymbol{M} \in \{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}\}, \forall \kappa \in H, \mathsf{z}_M(\kappa) = \sum_{\iota \in H} \boldsymbol{M}[\kappa, \iota]\mathsf{z}(\iota). \tag{8}$$

The ideal verifier checks the following equalities

$$\begin{aligned}
\mathsf{h}_3(\beta_3)\mathsf{Z}_K(\beta_3) &= \mathsf{a}(\beta_3) - \mathsf{b}(\beta_3)(\beta_3\mathsf{g}_3(\beta_3) + \sigma_3/|K|) \\
\mathsf{r}(\alpha, \beta_2)\sigma_3 &= \mathsf{h}_2(\beta_2)\mathsf{Z}_H(\beta_2) + \beta_2\mathsf{g}2(\beta_2) + \sigma_2/|H| \\
\mathsf{s}(\beta_1) + \mathsf{r}(\alpha, \beta_1)(\sum_M \eta_M \mathsf{z}_M(\beta_1)) - \sigma_2\mathsf{z}(\beta_1) &= \mathsf{h}_1(\beta_1)\mathsf{Z}_H(\beta_1) + \beta_1\mathsf{g}_1(\beta_1) + \sigma_1/|H| \\
\mathsf{z}_A(\beta_1)\mathsf{z}_B(\beta_1) - \mathsf{z}_C(\beta_1) &= \mathsf{h}_0(\beta_1)\mathsf{Z}_H(\beta_1)
\end{aligned} \tag{9}$$

where $\mathsf{g}_i(X), \mathsf{h}_i(X), i \in [1..3], \mathsf{a}(X), \mathsf{b}(X), \sigma_1, \sigma_2, \sigma_3$ are polynomials and variables required by the sumcheck protocol which allows the verifier to efficiently verify that Eq. (8) holds.

8.2 Unique Response Property

Lemma 12. *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 5 and Lemma 9. 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 $2 \cdot \varepsilon_{\mathsf{bind}}(\lambda) + \varepsilon_{\mathsf{op}}(\lambda)$. $\qquad\square$

8.3 Rewinding-Based Knowledge Soundness

Lemma 13. M_{FS} *is* $(2, 2n + 3)$-*rewinding-based knowledge sound against algebraic adversaries who make up to* q *random oracle queries with security loss*

$$\varepsilon_{\mathsf{ks}}(\lambda, \mathsf{acc}, q) \le \left(1 - \frac{\mathsf{acc} - (q + 1)\left(\frac{2n+2}{p}\right)}{1 - \frac{2n+2}{p}}\right) + (2n + 3) \cdot \varepsilon_{\mathsf{udlog}}(\lambda),$$

Here acc *is a probability that the adversary outputs an acceptable proof, and* $\varepsilon_{\mathsf{udlog}}(\lambda)$ *is the security of* $(2n+2, 1)$-udlog *assumption.*

Proof. The proof is similar to the proof of Lemma 6 and Lemma 10. We use Attema et al. [5, 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

$$1 - \frac{\mathsf{acc} - (q + 1)\left(\frac{2n+2}{p}\right)}{1 - \frac{2n+2}{p}}$$

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 $(2n + 3) \cdot \varepsilon_{\mathsf{udlog}}(\lambda)$. In the case that all the transcripts are accepting by the ideal verifier, but $\mathsf{Ext}_{\mathsf{ks}}$ 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. $\quad\square$

8.4 Trapdoor-Less Zero-Knowledge of Marlin

Lemma 14. 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.

8.5 Simulation Extractability of M_{FS}

Since Lemmas 12 to 14 hold, M_{FS} is 2-UR, rewinding-based knowledge sound and trapdoor-less zero-knowledge. By making use of Theorem 1, we conclude that M_{FS} is simulation-extractable as defined in Definition 2.

Corollary 3 (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*

$$\varepsilon_{\mathsf{se}}(\lambda, \mathsf{acc}, q) \leq \left(1 - \frac{\mathsf{acc} - \varepsilon_{\mathsf{ur}}(\lambda) - (q+1)\varepsilon_{\mathsf{err}}(\lambda)}{1 - \varepsilon_{\mathsf{err}}(\lambda)}\right) + (2n + 3) \cdot \varepsilon_{\mathsf{udlog}}(\lambda),$$

where $\varepsilon_{\mathsf{err}}(\lambda) = \frac{2n+2}{p}$, *p is the size of the field, and* n *is the number of constrains in the circuit.*

References

1. B. Abdolmaleki, K. Baghery, H. Lipmaa, and M. Zajac. A subversion-resistant SNARK. In T. Takagi and T. Peyrin, editors, *ASIACRYPT 2017, Part III*.
2. B. Abdolmaleki, S. Ramacher, and D. Slamanig. Lift-and-shift: Obtaining simulation extractable subversion and updatable SNARKs generically. In J. Ligatti, X. Ou, J. Katz, and G. Vigna, editors, *ACM CCS 2020*.
3. S. Atapoor and K. Baghery. Simulation extractability in groth's zk-SNARK. Cryptology ePrint Archive, Report 2019/641, 2019. https://eprint.iacr.org/2019/641.
4. T. Attema, S. Fehr, and M. Klooß. Fiat-shamir transformation of multi-round interactive proofs. Cryptology ePrint Archive, Report 2021/1377.
5. T. Attema, S. Fehr, and M. Klooß. Fiat-shamir transformation of multi-round interactive proofs. Cryptology ePrint Archive, Report 2021/1377, 2021. https://eprint.iacr.org/2021/1377.
6. K. Baghery, M. Kohlweiss, J. Siim, and M. 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.
7. M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In A. Juels, R. N. Wright, and S. De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, Oct. / Nov. 2006.
8. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993.
9. M. Ben-Or, O. Goldreich, S. Goldwasser, J. Håstad, J. Kilian, S. Micali, and P. Rogaway. Everything provable is provable in zero-knowledge. In S. Goldwasser, editor, *CRYPTO '88*, volume 403 of *LNCS*, pages 37–56. Springer, Heidelberg, Aug. 1990.
10. E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018. https://eprint.iacr.org/2018/046.
11. E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014.
12. E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 90–108. Springer, Heidelberg, Aug. 2013.
13. E. Ben-Sasson, A. Chiesa, and N. Spooner. Interactive oracle proofs. In M. Hirt and A. D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Heidelberg, Oct. / Nov. 2016.

14. E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In K. Fu and J. Jung, editors, *USENIX Security 2014*, pages 781–796. USENIX Association, Aug. 2014.

15. N. Bitansky, A. Chiesa, Y. Ishai, R. Ostrovsky, and O. Paneth. Succinct non-interactive arguments via linear interactive proofs. In A. Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 315–333. Springer, Heidelberg, Mar. 2013.

16. J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357. Springer, Heidelberg, May 2016.

17. S. Bowe and A. 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.

18. G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *Journal of computer and system sciences*, 37(2):156–189, 1988.

19. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.

20. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups (extended abstract). In B. S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 410–424. Springer, Heidelberg, Aug. 1997.

21. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. https://eprint.iacr.org/2000/067.

22. M. Chase and A. Lysyanskaya. On signatures of knowledge. In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, Heidelberg, Aug. 2006.

23. A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Heidelberg, May 2020.

24. G. Danezis, C. Fournet, J. Groth, and M. Kohlweiss. Square span programs with applications to succinct NIZK arguments. In P. Sarkar and T. Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 532–550. Springer, Heidelberg, Dec. 2014.

25. Y. Dodis, K. Haralambiev, A. López-Alt, and D. Wichs. Efficient public-key cryptography in the presence of key leakage. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 613–631. Springer, Heidelberg, Dec. 2010.

26. S. Faust, M. Kohlweiss, G. A. Marson, and D. Venturi. On the non-malleability of the Fiat-Shamir transform. In S. D. Galbraith and M. Nandi, editors, *INDOCRYPT 2012*, volume 7668 of *LNCS*, pages 60–79. Springer, Heidelberg, Dec. 2012.

27. M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 152–168. Springer, Heidelberg, Aug. 2005.

28. L. Fortnow. The complexity of perfect zero-knowledge (extended abstract). In A. Aho, editor, *19th ACM STOC*, pages 204–209. ACM Press, May 1987.

29. G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, Aug. 2018.

30. A. Gabizon, Z. J. Williamson, and O. 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.

31. C. Ganesh, C. Orlandi, M. Pancholi, A. Takahashi, and D. Tschudi. Fiat–shamir bulletproofs are non-malleable (in the algebraic group model). Cryptology ePrint Archive, Report 2021/1393, 2021. https://ia.cr/2021/1393.

32. R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct NIZKs without PCPs. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013.

33. Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. Cryptology ePrint Archive, Report 2017/454, 2017. https://eprint.iacr.org/2017/454.

34. O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th FOCS*, pages 174–187. IEEE Computer Society Press, Oct. 1986.

35. S. Goldwasser and Y. T. Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th FOCS*, pages 102–115. IEEE Computer Society Press, Oct. 2003.

36. J. Groth. Fully anonymous group signatures without random oracles. In K. Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 164–180. Springer, Heidelberg, Dec. 2007.

37. J. Groth. Short pairing-based non-interactive zero-knowledge arguments. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, Dec. 2010.

38. J. Groth. On the size of pairing-based non-interactive arguments. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.

39. J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn, and I. Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Heidelberg, Aug. 2018.

40. J. Groth and M. Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 581–612. Springer, Heidelberg, Aug. 2017.

41. A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Heidelberg, Dec. 2010.

42. A. Kosba, Z. Zhao, A. Miller, Y. Qian, H. Chan, C. Papamanthou, R. Pass, a. shelat, and E. Shi. How to use SNARKs in universally composable protocols. Cryptology ePrint Archive, Report 2015/1093, 2015. `https://eprint.iacr.org/2015/1093`.

43. H. Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In R. Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Heidelberg, Mar. 2012.

44. H. Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In K. Sako and P. Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 41–60. Springer, Heidelberg, Dec. 2013.

45. H. Lipmaa. Key-and-argument-updatable QA-NIZKs. Cryptology ePrint Archive, Report 2019/333, 2019. `https://eprint.iacr.org/2019/333`.

46. M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In L. Cavallaro, J. Kinder, X. Wang, and J. Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, Nov. 2019.

47. S. Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, Nov. 1994.

48. J. 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-bulletproofs-` 2022.

49. B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013.

50. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000.

51. L. Rotem and G. Segev. Tighter security for schnorr identification and signatures: A high-moment forking lemma for Σ-protocols. In T. Malkin and C. Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 222–250, Virtual Event, Aug. 2021. Springer, Heidelberg.