On Non-Malleability of Fiat–Shamir Based Universal zkSNARKs

(Submission to EUROCRYPT 2022)

Abstract. The Fiat–Shamir transformation turns public-coin (three round) sigma protocol into signature schemes, non-interactive proof systems, and signatures of knowledge (SoK). The security of the transformation relies on a powerful forking lemma that extracts 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 extend this line of work and formally define simulation extractability for protocols in the random oracle model (ROM) which use a structured reference string (SRS). We then show sufficient conditions for compiling via the Fiat–Shamir transformation public-coin multi-round interactive protocols with SRS into simulation-extractable NIZK proof systems. We also consider the case that 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. In the ROM, we obtain simulation-extractable and updatable NIZKs. Importantly, we show that thee popular zero knowledge SNARKs — Plonk, Sonic, and Marlin — are simulation extractable out-of-the-box. This also results in the first construction of updatable simulation-extractable SNARKs and succinct updatable SoK.

1 Introduction

Zero-knowledge proof systems that allow a prover to convince a verifier of a statement without revealing anything beyond the truth of the statement have applications in cryptography and theory of computation [8,23,28]. When restricted to computationally sound proofs, called *argument systems*, proof length can be shorter than the length of the witness [**?**]. Zero-knowledge Succinct Non-interactive ARguments of Knowledge (zkSNARKs) are zero-knowledge argument systems that additionally have a succinctness property – small proof sizes and fast verification. Since their introduction in [43], zk-SNARKs have been a powerful and versatile design tool for secure cryptographic protocols. They became particularly relevant for blockchain applications that demand short proofs and fast verification, such as privacy-preserving cryptocurrencies [9] in Zcash and scalable and private smart contracts in Ethereum[1].

The work of [26] proposed a preprocessing zk-SNARK for general NP statements phrased in the language of Quadratic Span Programs (QSP) and Quadratic Arithmetic Programs (QAP) for Boolean and arithmetic circuits respectively. This built on previous works of [**?**, 31, 39] and led to several works [10, 12, 13, 32, 40, 44] with very short proof sizes and fast verification. The line of work on pre-processing zkSNARKs has seen rapid progress with many works proposing significant improvements in efficiency of different parameters like proof size, verifier efficiency, complexity of setup etc.

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

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), that is then compiled via cryptographic tools to obtain an interactive argument system. This is then turned into a zkSNARK using the Fiat-Shamir transformation in the Random Oracle Model (ROM). In particular, several schemes such as Sonic in [42], Plonk [25], Marlin [18] follow this approach where the information theoretic object is an algebraic variant of IOP, and the cryptographic primitive in the compiler is a polynomial commitment scheme (PC).

Simulation extractability. Most zkSNARKs are shown to satisfy a standard knowledge soundness property. Intuitively, this guarantees that a prover that creates a valid proof knew a valid witness. However, deployments of zkSNARKs in real-world applications require a stronger property – *simulation-extractability* (SE). This is because, in practice, an adversary against the zkSNARK has access to proofs provided by other parties using the same zkSNARK. The definition of knowledge soundness 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 the chain for all blockchain-participants to see. Therefore, it is necessary for a zero-knowledge proof system to be *non-malleable*, that is, resilient against adversaries that additionally get to see proofs generated by different parties before trying to forge. Therefore, it is necessary for a zero-knowledge proof system to be *simulation-extractable*, that is, resilient against adversaries that additionally get to see proofs generated by different parties before trying to forge. This captures the more general scenario where an adversary against the zkSNARK has access to proofs provided by other parties as it is in applications of zkSNARKs in privacy-preserving blockchains, where proofs are posted on the chain for all participants in the network to verify.

zkSNARKs in the updatable setting. One of the downsides of efficient zkSNARKs like [19, 26, 31, 32, 39, 40, 44] 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, then they can produce proofs of false statement. That is, if the trusted setup assumption does not hold, knowledge soundness breaks down. Groth et al [33] propose a setting to tackle this challenge which allows parties – provers and verifiers – to *update* the SRS, that is, take a current SRS and contribute to it 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 who updates the SRS is honest. The SRS is also *universal*, in that it does not depend on the relation to be proved but only an upper bound on the size of the statements. Although inefficient, as the SRS length is quadratic to the size of the statement, [33] set a new paradigm of universal updatable setting for designing zkSNARKs.

The first universal zkSNARK with updatable and linear size SRS was Sonic proposed by Maller et al. in [42]. Subsequently, Gabizon et al. designed Plonk [25] which currently is the most efficient updatable universal zkSNARK. Independently, Chiesa et al. [18] 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 this updatable setting. Consider the following scenario: an instance proof pair with respect to some SRS is available for public verification, (srs, x, π). Now, if there is a new purported proof (srs', x, π') with respect to an updated srs', we would like the guarantee that the prover must have known a witness corresponding to x, and therefore computed π'. Since everybody is allowed to update an SRS, the ability for an adversary to perform an update srs to srs', and "move" the proof π from the old SRS to a proof π' for the new SRS without knowing a witness clearly violates security. That is, even an adversary who knows the trapdoor for the update from srs to srs' should not be able to break SE as long as there was at least one honest update to srs.

As it turns out, defining SE for updatable SRS zkSNARKs requires some care. Since the SRS is being continually updated, there are proofs with respect to *different* SRSs available for the adversary to see before attempting to forge a proof with respect to a current SRS. That is, each SRS in the update chain spawns a simulation oracle. Intuitively, the updatability of the SRS allows an adversarial prover to contribute to updating, and see proofs with respect to different updated SRSs before attempting to provide a proof for a false statement (potentially output a proof wrt a SRS that is different from the SRSs corresponding to all the simulated proofs seen). A definition of SE in the updatable setting should take into account this additional power of the adversary, which is not captured by existing definitions of SE. While generic lifting techniques/compilers [1, 38] 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.

Fiat–Shamir. The Fiat–Shamir (FS) transform takes a public-coin interactive protocol and makes it interactive by hashing the current protocol transcript to compute the verifier's public coins. While in principle justifiable in the random oracle model (ROM) [7], it is theoretically unsound [29] and so only a heuristic that should be used with care. The FS transform is a now popular design tool in constructing zkSNARKs. In the updatable universal SRS setting, works like Sonic [42] Plonk [25], and Marlin [18] are designed and proven secure as multi-round interactive protocols. Security is then only *conjectured* for their non-interactive variants by employing the FS transform.

We investigate the non-malleability properties of a class of zkSNARK protocols obtained by FS-compiling multi-round protocols in the updatable SRS setting and give a modular approach to analyze non-malleablilty of zkSNARKs.

1.1 Our 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 having access to updating the SRS and seeing proofs with respect to different SRSs.Michal 28.09: Now the adversary sees additional proofs wrt to the final SRS.
- *General theorem for USE of FS-compiled interactive protocols.* We then show that a class of interactive proofs of knowledge that are trapdoor-less simulatable, have a unique response property in the updatable setting, and satisfy a property we define

called forking soundness *are USE out-of-the box* in the random oracle model when the Fiat–Shamir transformation is applied to them. Informally, our notion of forking soundness is a variant of special soundness where the transcripts provided to the extractor are obtained through interaction with an honest verifier, and the extraction guarantee is computational instead of unconditional. Our extractor only needs oracle access to the adversary, it does not depend on the adversary's code, nor relies on knowledge assumptions.

- *USE for concrete zkSNARKs.* We then prove that the most efficient updatable SRS SNARKS – Plonk/Sonic/Marlin – satisfy the notions necessary to invoke our general theorem, thus showing that these zkSNARKs are updatable simulation extractable. Proving that these protocols satisfy the required properties is done in the algebraic group model (AGM).

1.2 Technical Overview

The proof of our general theorem for USE is, at a high level, along the lines of the proof of SE for FS-compiled sigma protocol from [21]. However, we need new and stronger notions as we consider proof systems that are richer than simple sigma protocols **[Hamid:** and moreover they are in the stronger updatable setting]. 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 by the Fiat–Shamir transform. In the following, we denote the underlying interactive protocols by P (for Plonk), S (for Sonic), and M (for Marlin) and the resulting non-interactive proof systems by P_{FS}, S_{FS}, M_{FS} respectively.

Forking soundness in the updatable setting. Following [21], 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. However, many protocols do not meet the standard definition of special soundness for sigma protocols, that requires extraction of a witness from any two transcripts, with the same first message. We put forth a notion analogous to special soundness, that is more general and applicable to a wider class of protocols – protocols that are more than three rounds, and rely on an updatable SRS. For P, S, and M that are not three move protocols, the definition needs to be adapted. Furthermore, the number of transcripts required for extraction is more than two. Concretely, $(3\mathsf{n} + 1)$ – where n is the number of constraints in the proven circuit – for Plonk, $(\mathsf{n} + \mathsf{Q} + 1)$ – where n and Q are the numbers of multiplicative and linear constraints – for Sonic, and $(\mathsf{n} + 3)$ – where n is the number of multiplicative constraints – for Marlin. Hence, we do not have a pair of transcripts, but a *tree of transcripts*.

In protocols that rely on SRS that come with a trapdoor, an adversary in possession of the trapdoor can produce multiple valid proof transcripts without knowing the witness and potentially for false statements. This is true even in the updatable setting, where there exists a trapdoor for any updated SRS. Recall that the standard special soundness definition requires witness extraction from *any* tree of acceptable transcripts that share a common root. This means that there are no such trees for false statements. We define a different, forking lemma-related, version of soundness that we call forking soundness. Forking 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, given that the trees are generated as interactions between a (possibly malicious) prover and an honest verifier. That is, if extraction from such a tree fails, then we break an underlying computational assumption.[**Hamid:** The fact that the definition is in the updatable setting indicates that this should hold even against adversaries that contribute in the SRS generation.]

Unique response property in the updatable setting. Another property required to show USE is the unique response property [22] which says that for 3-messages sigma protocols, all but the first message sent by the prover are deterministic (intuitively, the prover can only employ fresh randomness in the first message of the protocol). We cannot use this definition as is since the protocols we consider have other rounds where the prover messages are randomized. In Plonk, both the first and the second prover's messages are randomized. Although Sonic prover is deterministic after it picks its first message, the protocol has more than 3 rounds. The same holds for Marlin. We propose a generalisation of the definition which states that a protocol is i-ur if the prover is deterministic starting from its $(i + 1)$-th message. For our proof it is sufficient that this property is met by Plonk for $i = 2$. Since Sonic and Marlin provers are deterministic from the second message on, they are 1-ur.

Trapdoor-less simulatable protocols. In order to invoke our main theorem on (non-interactive variants of) Plonk, Sonic and Marlin to conclude USE, we also need to show that simulators in these protocols produces proofs without relying on the knowledge of trapdoor. More precisely for our reduction, we need simulators that rely only on reordering the messages and picking suitable verifier challenges, without knowing the SRS trapdoor. That is, any PPT party should be able to produce a simulated proof by its own in a trapdoor-less way. Note that this property does not necessary break soundness of the protocol as the simulator is required only to produce a transcript and is not involved in a real conversation with a real verifier. We show simulators for P_{FS}, S_{FS}, and M_{FS} that rely only on the programmability of the RO, where programmability is only needed from some round i onwards. [**Chaya:** revisit this. is HVZK for the interactive protocol? then why programming? it might be a good idea to elaborate on why a trapdoor-based simulator does not work in the reduction. I am not sure I have clarity on this.] Michal 16.09: maybe we should just define Trapdoor-less simulatable (TLS) protocols?Michal 28.09: some re-writing to incorporate TLS. check

Generalisation of the general forking lemma. Consider an interactive 3-message special-sound protocol Ψ and its non-interactive version Ψ_{FS} obtained by the Fiat–Shamir transform. The general forking lemma provides an instrumental lower bound for the probability of extracting a witness from an adversary who provides two proofs for the same statement that share the first message. Since P and S have more than 3 messages and are not special-sound, the forking lemma of Bellare and Neven [6], cannot be used directly. We propose a generalization that covers multi-message protocols where witness extraction requires more transcripts than merely two. Unfortunately, we also observe that the security gap grows with the number of transcripts and the probability that the extractor succeeds diminishes significantly; the security loss, albeit big, is polynomial.

Most modern zkSNARKs [16, 42] heavily rely on the Fiat–Shamir transform and thus potentially the forking lemma. First, an interactive protocol is proposed and its security and forking soundness analysed. Second, one uses an argument that the Fiat–

Shamir transform can be used to get a protocol that is non-interactive and shares the same security properties.

We see our generalized forking lemma as contributing to a critical assessment of this approach. The analysis of the interactive protocol is not enough and one has to consider the security loss implied by the Fiat-Shamir transform for the target security notion. Thus one has to either rely on our generalisation of the forking lemma or disclose a transformation that does not suffer this loss. We note that the security loss may also apply when knowledge soundness is proven. That is the case for the original Sonic paper, whose security proof relies on so-called witness-extended emulation. The authors of Plonk and recent work on Sonic [27] work around this problem by proving knowledge soundness directly in the AGM.

1.3 Related Work

Simulation extractability. There are many results on simulation extractability for non-interactive zero-knowledge proofs (NIZKs). First, Groth [30] noticed that a (black-box) SE NIZK is universally-composable (UC) [17]. Then Dodis et al. [20] introduced a notion of (black-box) *true simulation extractability* 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 [34] and SE zkSNARK for QAP by Lipmaa [41]. Kosba's et al. [38] give a general transformation from a NIZK to a black-box SE NIZK. Although their transformation works for zkSNARKs as well, succinctness of the proof system is not preserved by the transformation. Recently, Abdolmaleki et al. [1] showed another transformation that obtains non-black-box simulation extractability but also preserves succinctness of the argument. The zkSNARK of [32] has been shown to be SE by introducing minor modifications to the construction and making stronger assumptions [2,15]. Recently, [3] showed that the original Groth's proof system from [32] is weakly SE and randomizable. None of these results are for zkSNARKs in the updatable SRS setting.

Forking lemma generalizations. There are several task specific variants, e.g., [4,5,35], of the general forking lemma [6,45] for analyzing the forking behavior of random-oracle based executions. In [14], Bootle et al. proposed a novel inner-product argument which security relies on, so-called, witness-extended emulation. To show that property, the authors proposed a new version of forking lemma, which gives a lower bound on probability that a tree finding algorithm is able to produce a tree of acceptable transcripts by rewinding a conversation between a (potentially malicious) prover and verifier.

Although the result in that paper is dubbed a "forking lemma" it differs from forking lemmas known from e.g. [6,45]. First of all, the forking lemmas in these papers analyse the probability of building a tree of acceptable transcripts for Fiat–Shamir based non-interactive proof systems, while the protocol presented by Bootle et al. is intended to work for interactive proof systems.

Importantly, it is not obvious how the result of Bootle et al. can be used to show security of non-interactive protocols as it relies on interactive provers whose proving strategies are more limited than proving strategies of non-interactive provers. For example, if a challenge given by the verifier does not suit an interactive prover, it can only try to finish a proof with it or abort. On the other hand, a non-interactive prover has far wider scope of possible actions–when the protocol is non-interactive the prover may adapt its strategy based on the random oracle outputs. This is reminiscent of *state restoration* security [11, 36] which is also about the security loss incurred by FS transformation for knowledge soundness from witness extended emulation.

Here, we directly capture the state restoration capability of the prover in the forking lemma instead of defining an interactive game where the prover can rewind the verifier to an earlier state as is done in [**?**]. The work of [**?**] further shows that state restoration security gives tight security guarantees for the non-interactive versions of Bulletproof [16] and Sonic . Our work differs from [**?**] in the following ways. First, they focus on showing security of concrete proof systems, while we show a general theorem about the security of a wide class of protocols. Second, they only consider knowledge soundness, while we focus on the stronger notion of simulation extractability. Third, the proof of [**?**] is in the AGM which allows for online extraction, whereas we aim to minimize our reliance on the AGM. In particular, our main theorem does not rely on AGM and we tackle technical challenges arising from extraction by rewinding. However, note that we show that concrete protocols satisfy the preconditions of our main theorem in the AGM.

2 Preliminaries

Notation. Let PPT denote probabilistic polynomial-time and $\lambda \in \mathbb{N}$ be the security parameter. All adversaries are stateful. For an algorithm $\mathcal{A}$, let $\mathrm{im}(\mathcal{A})$ be the image of $\mathcal{A}$ (the set of valid outputs of $\mathcal{A}$), let $\mathsf{R}(\mathcal{A})$ denote the set of random tapes of correct length for $\mathcal{A}$ (assuming the given value of λ), and let $r \leftarrow\!\!\$\ \mathsf{R}(\mathcal{A})$ denote the random choice of the randomiser r from $\mathsf{R}(\mathcal{A})$. We denote by $\mathsf{negl}(\lambda)$ ($\mathsf{poly}(\lambda)$) an arbitrary negligible (resp. polynomial) function.

Probability ensembles $X = \{X_\lambda\}_\lambda$ and $Y = \{Y_\lambda\}_\lambda$, for distributions X_λ, Y_λ, have *statistical distance* Δ equal $\epsilon(\lambda)$ if $\sum_{a \in \mathrm{Supp}(X_\lambda \cup Y_\lambda)} |\Pr[X_\lambda = a] - \Pr[Y_\lambda = a]| = \epsilon(\lambda)$. We write $X \approx_\lambda Y$ if $\Delta(X_\lambda, Y_\lambda) \le \mathsf{negl}(\lambda)$. For values $a(\lambda)$ and $b(\lambda)$ we write $a(\lambda) \approx_\lambda b(\lambda)$ if $|a(\lambda) - b(\lambda)| \le \mathsf{negl}(\lambda)$.

For a probability space $(\Omega, \mathcal{F}, \mu)$ and event $\mathsf{E} \in \mathcal{F}$ we denote by $\overline{\mathsf{E}}$ an event that is complementary to E, i.e. $\overline{\mathsf{E}} = \Omega \setminus \mathsf{E}$.

[**Chaya:** revisit after agreeing on how we deal with relations.] Denote by $\mathcal{R} = \{\mathbf{R}\}$ a family of relations. We assume that if $\mathbf{R}$ comes with any auxiliary input, the latter is benign. Directly from the description of $\mathbf{R}$ one learns security parameter λ and description of the group $\mathbb{G}$, if the relation is a relation of group elements (as it usually is in case of zkSNARKs).

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 ag_ι where g_ι is a fixed generator of $\mathbb{G}_\iota$. We denote $\hat{e}([a]_1, [b]_2)$ as $[a]_1 \bullet [b]_2$. Thus, $[a]_1 \bullet [b]_2 = [ab]_T$. We freely use the bracket notation with matrices, e.g., if $\boldsymbol{AB} = \boldsymbol{C}$ then $\boldsymbol{A}\, [\boldsymbol{B}]_\iota = [\boldsymbol{C}]_\iota$ and $[\boldsymbol{A}]_1 \bullet [\boldsymbol{B}]_2 = [\boldsymbol{C}]_T$. 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 with generating these parameters by Pgen.

Lemma 1 (Difference lemma, [47, Lemma 1]). *Let* $\mathsf{A}, \mathsf{B}, \mathsf{F}$ *be events defined in some probability space, and suppose that* $\mathsf{A} \wedge \overline{\mathsf{F}} \iff \mathsf{B} \wedge \overline{\mathsf{F}}$. *Then* $|\Pr[\mathsf{A}] - \Pr[\mathsf{B}]| \leq \Pr[\mathsf{F}]$.

2.1 Algebraic Group Model

The algebraic group model (AGM) introduced in [24] lies between the standard model 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 that 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 $[y] = \boldsymbol{c}^\top \cdot [\boldsymbol{x}]$. Both Plonk and Sonic have been shown secure using the AGM. An adversary that works in the AGM is called *algebraic*.

2.2 Polynomial commitment

In the polynomial commitment scheme $\mathsf{PC} = (\mathsf{KGen}, \mathsf{Com}, \mathsf{Op}, \mathsf{Vf})$ the committer C can convince the receiver R that some polynomial f which C committed to evaluates to s at some point z chosen by R. PC's subroutines are defined as follows

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

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

$\mathsf{Vf}(\mathsf{srs}, c, z, s, o)$**:** 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.

Plonk and Sonic use variants of the KZG polynomial commitment scheme [37]. We denote the first by $\mathsf{PC_P}$ and the latter by $\mathsf{PC_S}$. Due to page limit, we omit their presentation here and refer to Fig. 3 and Fig. 4 in the Section 6. In this paper we use evaluation binding, commitment of knowledge, and, newly introduced, unique opening and hiding properties. Formal definitions of these could be find in Section 6, here we briefly introduce them.

Evaluation binding intuitively, this property assures that no adversary could provide two valid openings for two different evaluations of the same commitment in the same point.

Commitment of knowledge when a commitment scheme is "of knowledge" then if an adversary produces a (valid) commitment c, which it can open, then it also knows the underlying polynomial f which commits to that value. [42] shows, using AGM, that $\mathsf{PC_S}$ is a commitment of knowledge. The same reasoning could be used to show that property for $\mathsf{PC_P}$.

Unique opening this property assures that there is only one valid opening for the committed polynomial and given evaluation point. This property is crucial in showing forking simulation-extractability of Plonk and Sonic. We show that the Plonk's and Sonic's polynomial commitment schemes satisfy this requirement in Lemma 3 and **??** respectively.

Hiding assures that no adversary is able to tell anything about the polynomial given only its commitment and bounded number of evaluations.

2.3 Zero-Knowledge Proof Systems

Let $\mathcal{R}(1^\lambda) = \{\mathbf{R}\}$ be a family of NP relations. Denote by $\mathcal{L}_\mathbf{R}$ the language determined by $\mathbf{R}$. Let P be a *prover* and V be the *verifier*, both PPT algorithms. We allow our proof system to have a setup, i.e. there is a KGen algorithm that takes as input the relation description $\mathbf{R}$ and outputs a common reference string srs. We assume that the srs defines the relation and for universal prove systems, such as Plonk and Sonic, we treat both the reference string and the relation as universal.

We denote by $\langle \mathsf{P}(\mathsf{srs}, \mathsf{x}, \mathsf{w}), \mathsf{V}(\mathsf{srs}, \mathsf{x}) \rangle$ a *transcript* (also called *proof*) π of a conversation between P with input $(\mathsf{srs}, \mathsf{x}, \mathsf{w})$ and V with input $(\mathsf{srs}, \mathsf{x})$. We write $\langle \mathsf{P}(\mathsf{srs}, \mathsf{x}, \mathsf{w}), \mathsf{V}(\mathsf{srs}, \mathsf{x}) \rangle = 1$ if in the end of the transcript the verifier V returns 1 and say that V accepts it. For non-interactive proof systems we abuse notation and write $\mathsf{V}(\mathsf{srs}, \mathsf{x}, \pi) = 1$ to denote a fact that π is accepted by the verifier.

A proof system $\Psi = (\mathsf{KGen}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ for $\mathcal{R}$ is required to have three properties: completeness, soundness and zero knowledge, which are defined as follows:

Completeness. An interactive proof system Ψ is *complete* if an honest prover always convinces an honest verifier, that is for all $\mathbf{R} \in \mathcal{R}(1^\lambda)$ and $(\mathsf{x}, \mathsf{w}) \in \mathbf{R}$

$$\Pr[\langle \mathsf{P}(\mathsf{srs}, \mathsf{x}, \mathsf{w}), \mathsf{V}(\mathsf{srs}, \mathsf{x}) \rangle = 1 \mid \mathsf{srs} \leftarrow \mathsf{KGen}(\mathbf{R})\,] = 1\,.$$

Soundness. We say that Ψ for $\mathcal{R}$ is *sound* if no PPT prover $\mathcal{A}$ can convince an honest verifier V to accept a proof for a false statement $\mathsf{x} \notin \mathcal{L}$. More precisely, for all $\mathbf{R} \in \mathcal{R}(1^\lambda)$

$$\Pr[\langle \mathcal{A}(\mathsf{srs}, \mathsf{x}), \mathsf{V}(\mathsf{srs}, \mathsf{x}) \rangle = 1 \land \mathsf{x} \notin \mathcal{L}_\mathbf{R} \mid \mathsf{srs} \leftarrow \mathsf{KGen}(\mathbf{R}), \mathsf{x} \leftarrow \mathcal{A}(\mathsf{srs})\,] \le \mathsf{negl}(\lambda).$$

Sometimes a stronger notion of soundness is required—except requiring that the verifier rejects proofs of statements outside the language, we request from the prover to know a witness corresponding to the proven statement. This property is called *knowledge soundness*.

Zero knowledge. We call a proof system Ψ *zero-knowledge* if for any $\mathbf{R} \in \mathcal{R}(1^\lambda)$, and adversary $\mathcal{A}$ there exists a PPT simulator Sim such that for any $(\mathsf{x}, \mathsf{w}) \in \mathbf{R}$

$$\left\{\langle \mathsf{P}(\mathsf{srs}, \mathsf{x}, \mathsf{w}), \mathcal{A}(\mathsf{srs}, \mathsf{x}, \mathsf{w}) \rangle \,\middle|\, \mathsf{srs} \leftarrow \mathsf{KGen}(\mathbf{R}) \right\} \approx_\lambda \left\{\mathsf{Sim}^{\mathcal{A}}(\mathsf{srs}, \mathsf{x}) \,\middle|\, \mathsf{srs} \leftarrow \mathsf{KGen}(\mathbf{R}) \right\}.$$

We call zero knowledge *perfect* if the distributions are equal and *computational* if they are indistinguishable for any PPT distinguisher.

Alternatively, zero-knowledge can be defined by allowing the simulator to use the trapdoor td that is generated along the srs. In this paper we distinguish simulators that requires a trapdoor to simulate and those that do not. We call the former *SRS-simulators*.

We say that a protocol is zero knowledge in the standard model if its simulator does not require the trapdoor.

In security reductions in this paper it is sometimes needed to produce simulated NIZK proofs without knowning the trapdoor, just by programming the random oracle. We call protocols which allow for such kind of simulation *trapdoor-less simulatable* (TLS). More precisely,

Definition 1 (Trapdoor-Less Simulatable Proof System). *Let* $\Psi = (\mathsf{KGen}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *be a NIZK proof system and* $\mathcal{H}$ *a random oracle. Let* Sim *be a pair of algorithms:* $\mathsf{Sim}_{\mathcal{H}}$ *that takes random oracle queries and answers them,* $\mathsf{Sim}_{\mathsf{P}}$ *that takes as input an SRS* srs *and instance* x *and outputs a proof* π_{Sim}*. We call* Ψ trapdoor-less simulatable *if for any adversary* $\mathcal{A}$*,* $\varepsilon_0 \approx \varepsilon_1$*, where*

$$\varepsilon_b = \Pr\left[\mathcal{A}^{\mathsf{O}_b}(\mathsf{srs}) = 0 \mid \mathsf{srs} \leftarrow\!\!\$\ \mathsf{KGen}(\lambda)\right] \tag{1}$$

where O_b *takes two types of adversary's queries:*

random oracle calls: *on* $\mathcal{A}$*'s query* x*,* O_b *responds with* $\mathcal{H}(x)$ *if* $b = 0$*, and with* $y \leftarrow \mathsf{Sim}_{\mathcal{H}}(\mathsf{srs}, x)$*, if* $b = 1$*.*

proof calls: *on* $\mathcal{A}$*'s query* x, w *responds with a real proof* $\pi_{\mathsf{P}} \leftarrow \mathsf{P}(\mathsf{srs}, \mathsf{x}, \mathsf{w})$ *if* $b = 0$ *or a simulated proof* $\pi_{\mathsf{Sim}} \leftarrow \mathsf{Sim}(\mathsf{srs}, \mathsf{x})$ *if* $b = 1$*.*

Remark 1 (TLS vs HVZK). We note that TLS notion is closely related to honest-verifier zero knowledge in the standard model. That is, if we consider an interactive proof system Ψ that is HVZK in the standard model then its Fiat–Shamir compiled version Ψ_{FS} is TLS. This comes as the simulator Sim in Ψ produces a valid simulated proof by picking verifier's challenges according to a predefined distribution and Ψ_{FS}'s simulator $\mathsf{Sim}_{\mathsf{FS}}$ produces its proofs similarly by picking the challenges and additionally programming the random oracle to return the picked challenges. Importantly, in both Ψ and Ψ_{FS} success of the simulator does not depend on access to an SRS trapdoor (which may not even exists if the proof systems are transparent).

Definition 2 (k-programmable ZK). *Let* Ψ *be a* $(2\mu + 1)$*-message ZK proof system and let* Ψ_{FS} *be its Fiat–Shamir variant. We say that* Ψ_{FS} *is* k*-programmable ZK if there exists a simulator* $\mathsf{Sim}_{\mathsf{FS}}$ *that*

1. *produces proofs indistinguishable from proofs output by an honest prover;*
2. $\mathsf{Sim}_{\mathsf{FS}}$ *programs the random oracle* only *for challenges from round* k *to* $\mu + 1$*.*

We note that Plonk is 2-programmable ZK, Sonic is 1-programmable ZK, and Marlin is 1-programmable ZK. This follows directly from the proofs of their standard model zero-knowledge property in Lemma 6 and **????**.

Idealised Verifier and Verification Equations Let $(\mathsf{KGen}, \mathsf{P}, \mathsf{V})$ be a proof system. Observe that the KGen algorithm provides an SRS which can be interpreted as a set of group representation of polynomials evaluated at trapdoor elements. E.g. for a trapdoor χ the SRS contains $[\mathsf{p}_1(\chi), \ldots, \mathsf{p}_\mathsf{k}(\chi)]_1$, for some polynomials $\mathsf{p}_1(X), \ldots, \mathsf{p}_\mathsf{k}(X) \in \mathbb{F}_p[X]$. On the other hand, the verifier V accepts if a (possibly set of) verification equation $\mathsf{ve}_{\mathsf{x},\pi}$ (note that the verification equation changes relate to the instance x and proof π), 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 [25] we call verifiers who checks that $\mathsf{ve}_{\mathsf{x},\pi}(\chi) = 0$ *real verifiers* as opposed to *ideal verifiers* who accepts 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

1. the idealised verifier accepts an incorrect proof (what "incorrect" means depends on the situation) with at most negligible probability (and many cases—never);
2. when the real verifier accepts, but not the idealised one, then we show how to use a malicious P to break the underlying security assumption (in our case—a variant of dlog.)

Analogously, idealised verifier can also be defined for polynomial commitment scheme.

Sigma protocols A sigma protocol $\Sigma = (\mathsf{P}, \mathsf{V}, \mathsf{Sim})$ for a relation $\mathbf{R} \in \mathcal{R}(1^\lambda)$ is a special case of an interactive proof where a transcript consists of three messages (a, b, z), where b is a challenge provided by the verifier. Sigma protocols are honest verifier zero-knowledge in the standard model and specially-sound. That is, there exists an extractor Ext which given two accepting transcripts $(a, b, z), (a, b', z')$ for a statement x can recreate the corresponding witness if $b \neq b'$. More formally:

Special soundness. A sigma protocol Σ is *specially-sound* if for any adversary $\mathcal{A}$ the probability is upper-bounded by some negligible function $\mathsf{negl}(\lambda)$:

$$\Pr\left[\begin{array}{l|l} \mathsf{V}(\mathbf{R}, \mathsf{x}, (a, b, z)) = \mathsf{V}(\mathbf{R}, \mathsf{x}, (a, b', z')) = 1 & (\mathsf{x}, (a, b, z), (a, b', z')) \leftarrow \mathcal{A}(\mathbf{R}), \\ \wedge\, b \neq b' \wedge \mathbf{R}(\mathsf{x}, \mathsf{w}) = 0 & \mathsf{w} \leftarrow \mathsf{Ext}(\mathbf{R}, \mathsf{x}, (a, b, z), (a, b', z')) \end{array}\right]$$

Another property that sigma protocols may have is a unique response property [22] which states that no PPT adversary can produce two accepting transcripts that differ only on the last element. More precisely:

Unique response property. Let $\Sigma = (\mathsf{P}, \mathsf{V}, \mathsf{Sim})$ be a sigma-protocol for $\mathbf{R} \in \mathcal{R}(1^\lambda)$ with proofs of the form (a, b, z). We say that Σ has the unique response property if for all PPT algorithms $\mathcal{A}$, it holds that the following probability is negligible ($\leq \mathsf{negl}(\lambda)$):

$$\Pr[\mathsf{V}(\mathbf{R}, \mathsf{x}, (a, b, z)) = \mathsf{V}(\mathbf{R}, \mathsf{x}, (a, b, z')) = 1 \wedge z \neq z' \mid (\mathsf{x}, a, b, z, z') \leftarrow \mathcal{A}(\mathbf{R})]$$

If this property holds even against unbounded adversaries, it is called *strict*, cf. [21]. Later on we call protocols that follows this notion *ur-protocols*. For the sake of completeness we note that many sigma protocols, like e.g. Schnorr's protocol [46], fulfil this property.

2.4 From interactive to non-interactive—the Fiat–Shamir transform

Consider a $(2\mu + 1)$-message, public-coin, honest verifier zero-knowledge interactive proof system $\Psi = (\mathsf{KGen}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ for $\mathbf{R} \in \mathcal{R}(1^\lambda)$. Let π be a proof performed by the prover P and verifier V compound of messages $(a_1, b_1, \ldots, a_\mu, b_\mu, a_{\mu+1})$, where a_i comes from P and b_i comes from V. Denote by $\mathcal{H}$ a random oracle. Let $\Psi_{\mathsf{FS}} = (\mathsf{KGen}_{\mathsf{FS}}, \mathsf{P}_{\mathsf{FS}}, \mathsf{V}_{\mathsf{FS}}, \mathsf{Sim}_{\mathsf{FS}})$ be a proof system such that

- $\mathsf{KGen}_{\mathsf{FS}}$ behaves as KGen.

- P_{FS} behaves as P except after sending message a_i, $i \in [1 \mathinner{..} \mu]$, the prover does not wait for the message from the verifier but computes it locally setting $b_i = \mathcal{H}(\pi[0..i])$, where $\pi[0..j] = (\mathsf{x}, a_1, b_1, \ldots, a_{j-1}, b_{j-1}, a_j)$. (Importantly, $\pi[0..\mu + 1] = (\mathsf{x}, \pi)$).
- V_{FS} behaves as V but does not provide challenges to the prover's proof. Instead it computes the challenges locally as P_{FS} does. Then it verifies the resulting transcript π as the verifier V would.
- $\mathsf{Sim}_{\mathsf{FS}}$ behaves as Sim, except when Sim picks challenge b_i before computing message $\pi[0, i]$, $\mathsf{Sim}_{\mathsf{FS}}$ programs the random oracle to output b_i on $\pi[0, i]$.

The Fiat–Shamir heuristic states that Ψ_{FS} is a zero-knowledge non-interactive proof system for $\mathbf{R} \in \mathcal{R}(1^\lambda)$.

2.5 Updatable SRS schemes

We recall the definition of an updatable SRS scheme from [33] which consists of the following algorithms.

- $(\mathsf{srs}, \rho) \leftarrow \mathsf{KGen}(1^\lambda)$ is a PPT algorithm that takes a security parameter λ and outputs a SRS srs, and correctness proof ρ.
- $(\mathsf{srs}', \rho') \leftarrow \mathsf{Upd}(1^\lambda, \mathsf{srs}, \{\rho_i\}_{i=1}^n)$ is a PPT algorithm that takes as input the security parameter λ, a SRS srs, a list of update proofs and outputs an updated SRS together with a proof of correct update.
- $b \leftarrow \mathsf{VerifySRS}(1^\lambda, \mathsf{srs}, \{\rho_i\}_{i=1}^n)$ is a DPT algorithm that takes the security parameter λ, a SRS srs, a list of update proofs, and outputs a bit indicating acceptance or not.

In the next section, we define security notions in the updatable setting. To this end, we define an SRS update oracle UpdO in Fig. 1 by which the adversary updates the SRS. We also define the simulation oracle SimO in Fig. 1 that is the simulator w.r.t. the SRS finalised by the adversary using UpdO. This simulation oracle will be used in the definition of forking simulation extractability.

[**Hamid:** TODO: add a paragraph here about the universality!]

3 Definitions and lemmas for multi-round SRS-based protocols

The result of Faust et al. [21] is for 3-round protocols that require two tramscripts for standard model extraction. Here, not only do we consider multi-round protocols, they require more than just two transcripts for extraction and are not special sound. Moreover, they rely on an SRS, and in the updatable setting, the adversary gets to contribute to the SRS in the updatable setting which makes the analysis more complicated. We first give a definition of simulation-extractability which is the one in [21] adapted to the updatable SRS setting. Then, we adapt special soundness to forking soundness, and generalize the forking lemma and the unique response property for multi-round SRS-based protocols.

Definition 3 (Forking simulation-extractable NIZK). *Let $\Psi = (\mathsf{KGen}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ be a NIZK proof system. We say that Ψ is* updatable forking simulation-extractable *with* extraction error ν *if for any* PPT *adversary $\mathcal{A}$ that is given oracle access to an updatable SRS setup* UpdO*, a simulation oracle* SimO*, cf. Fig. 1, and a random oracle $\mathcal{H}$, and*

```
$\mathsf{UpdO}(\mathtt{intent}, \mathsf{srs}_n, \{\rho_i\}_{i=1}^n)$
if $\mathsf{srs} \neq \bot$ : return $\bot$
if $(\mathtt{intent} = \mathtt{setup})$ :
    $(\mathsf{srs}', \rho') \leftarrow \mathsf{KGen}(\mathbf{R})$
    $Q_{\mathsf{srs}} \leftarrow Q_{\mathsf{srs}} \cup \{(\mathsf{srs}', \rho')\}$
    return $(\mathsf{srs}', \rho')$
if $(\mathtt{intent} = \mathtt{update})$ :
    $b \leftarrow \mathsf{VerifySRS}(1^\lambda, \mathsf{srs}_n, \{\rho_i\}_{i=1}^n)$
    if $(b = 0)$ : return $\bot$
    $(\mathsf{srs}', \rho') \leftarrow \mathsf{Upd}(1^\lambda, \mathsf{srs}_n, \{\rho_i\}_{i=1}^n)$
    $Q_{\mathsf{srs}} \leftarrow Q_{\mathsf{srs}} \cup \{(\mathsf{srs}', \rho')\}$
$//Q_{\mathsf{srs}} = (Q_{\mathsf{srs}}^{(1)}, Q_{\mathsf{srs}}^{(2)})$ s.t. $Q_{\mathsf{srs}}^{(2)}$ contains the update proofs in $Q_{\mathsf{srs}}$
    return $(\mathsf{srs}', \rho')$
if $(\mathtt{intent} = \mathtt{final})$ :
    $b \leftarrow \mathsf{VerifySRS}(1^\lambda, \mathsf{srs}_n, \{\rho_i\}_{i=1}^n)$
    if $(b = 0) \vee Q_{\mathsf{srs}}^{(2)} \cap \{\rho_i\}_i = \emptyset$ : return $\bot$
    $\mathsf{td} \leftarrow \mathsf{Ext}_{\mathsf{srs}}(\mathsf{srs}_n, , Q_{\mathsf{srs}}, \{\rho_i\}_{i=1}^n)$
    $\mathsf{srs} \leftarrow \mathsf{srs}_n$, return $\mathsf{srs}$
else return $\bot$
```

```
$\mathsf{SimO}(\mathsf{x}')$
if $(\mathsf{srs} = \bot)$ : return $\bot$
$\pi' \leftarrow$ [Hamid :$\mathsf{Sim}_{\mathsf{FS}}$]$(\mathsf{srs}, \mathsf{td}, \mathsf{x}')$
$Q = Q \cup \{(\mathsf{x}', \pi')\}$
return $\pi'$
```

Fig. 1: The left oracle defines the notion of updatable SRS setup. The right oracle is the simulation oracle.

produces an accepting transcript of Ψ *with probability* acc*, where*

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

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

$$\mathsf{ext} = \Pr\left[\begin{matrix} \mathsf{V}(\mathsf{srs}, \mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}) = 1 \wedge \\ (\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}) \notin Q \wedge \mathbf{R}(\mathsf{x}_{\mathcal{A}}, \mathsf{w}_{\mathcal{A}}) = 1 \end{matrix} \;\middle|\; \begin{matrix} r \leftarrow_{\$} \mathsf{R}(\mathcal{A}), (\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathsf{UpdO}, \mathsf{SimO}, \mathcal{H}}(1^\lambda; r) \\ \mathsf{w}_{\mathcal{A}} \leftarrow \mathsf{Ext}_{\mathsf{se}}(\mathsf{srs}, \mathcal{A}, r, \mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}, Q, Q_{\mathcal{H}}, Q_{\mathsf{srs}}) \end{matrix}\right]$$

is at at least

$$\mathsf{ext} \geq \frac{1}{\mathsf{poly}(\lambda)} (\mathsf{acc} - \nu)^d - \varepsilon(\lambda),$$

for some polynomial $\mathsf{poly}(\lambda)$*, constant* d *and negligible* $\varepsilon(\lambda)$ *whenever* $\mathsf{acc} \geq \nu$*. Here,* srs *is the finalized SRS, list* Q *contains all* (x, π) *pairs where* x *is an instance provided to the simulator by the adversary and* π *is the simulator's answer. List* $Q_{\mathcal{H}}$ *contains all* $\mathcal{A}$*'s queries to* $\mathcal{H}$ *and* $\mathcal{H}$*'s answers.*

3.1 Generalised forking lemma

Although dubbed "general", the forking lemma of **??** 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 $(3\mathsf{n} + 1)$ valid proofs, and $(\mathsf{n} + \mathsf{Q} + 1)$ for S. We propose a generalisation of the general forking lemma that given probability of producing an accepting transcript, acc, lower-bounds the probability of generating a *tree of accepting transcripts* T, which allows to extract a witness.

Definition 4 (Tree of accepting transcripts, cf. [14]). *Consider a $(2\mu + 1)$-message interactive proof system Ψ. A $(n_1, \dots, n_\mu)$-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 verifier's 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 $N = \mathsf{poly}(\lambda)$.*

Lemma 2 (General forking lemma II). *Fix $q \in \mathbb{Z}$ and set H of size $h \geq m$. Let $\mathcal{Z}$ be a PPT algorithm that on input $y, h_1, \dots, h_q$ returns (i, s) where $i \in [0 \mathinner{..} q]$ and s is called a side output. Denote by IG a randomised instance generator. We denote by acc the probability*

$$\Pr\left[i \neq 0 \,\middle|\, y \leftarrow \mathsf{IG};\ h_1, \dots, h_q \leftarrow\!\$\, H;\ (i, s) \leftarrow \mathcal{Z}(y, h_1, \dots, h_q)\right].$$

Let $\mathsf{GF}^m_{\mathcal{Z}}$ denote the algorithm described in Fig. 2 then the probability $\mathsf{frk} :=$ $\Pr\left[b = 1 \,\middle|\, y \leftarrow \mathsf{IG};\ h_1, \dots, h_q \leftarrow\!\$\, H;\ (b, \boldsymbol{s}) \leftarrow \mathsf{GF}^m_{\mathcal{Z}}(y, h_1, \dots, h_q)\right]$ *is at least*

$$\frac{\mathsf{acc}^m}{q^{m-1}} - \mathsf{acc} \cdot \left(1 - \frac{h!}{(h - m)! \cdot h^m}\right).$$

The proof is along similar lines as [6, Lemma 1] with modifications required by the fact that the protocol has more than 3 rounds and the number of transcripts required is larger. We defer the proof to **??**.

3.2 Unique-response protocols

Another technical hurdle is the assumption of unique response property of the transformed sigma protocol required by Faust et al. The original Fischlin's formulation, although suitable for applications presented in [21, 22], 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 rounds. Another problem is that the protocol could consist of a round other than the first one where the prover message 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 what also complicates things considerably.

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

```
$\mathsf{GF}^m_{\mathcal{Z}}(y, h^1_1, \ldots, h^1_q)$
$\rho \leftarrow\$ \mathsf{R}(\mathcal{Z})$
$(i, s_1) \leftarrow \mathcal{Z}(y, h^1_1, \ldots, h^1_q; \rho)$
$i_1 \leftarrow i$
**if** $i = 0$ **return** $(0, \bot)$
**for** $j \in [2 \mathinner{..} m]$
    $h^j_1, \ldots, h^j_{i-1} \leftarrow h^{j-1}_1, \ldots, h^{j-1}_{i-1}$
    $h^j_i, \ldots, h^j_q \leftarrow\$ H$
    $(i_j, s_j) \leftarrow \mathcal{Z}(y, h^j_1, \ldots, h^j_{i-1}, h^j_i, \ldots, h^j_q; \rho)$
    **if** $i_j = 0 \lor i_j \neq i$ **return** $(0, \bot)$
**if** $\exists (j, j') \in [1 \mathinner{..} m]^2, j \neq j' : (h^j_i = h^{j'}_i)$ **return** $(0, \bot)$
**else return** $(1, \boldsymbol{s})$
```

Fig. 2: Generalised forking algorithm $\mathsf{GF}^m_{\mathcal{Z}}$

Definition 5 (i-ur-protocol in the updatable setting). *Let Ψ be a $(2\mu + 1)$-message public coin proof system $\Psi = (\mathsf{KGen}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$. Let Ψ_{FS} be Ψ after the Fiat–Shamir transform and $\mathcal{H}$ the random oracle. Denote by $a_1, \ldots, a_\mu, a_{\mu+1}$ protocol messages output by the prover, We say that Ψ has* unique responses from i on *if for any* PPT *adversary $\mathcal{A}$:*

$$\Pr\left[\begin{array}{l|l} \boldsymbol{a} \neq \boldsymbol{a'}, a_1, \ldots, a_i = a'_1, \ldots, a'_i, & \mathsf{x}, \boldsymbol{a}, \boldsymbol{a'} \leftarrow \mathcal{A}^{\mathcal{H}, \mathsf{UpdO}}(1^\lambda) \\ \mathsf{V}^{\mathcal{H}}_{\mathsf{FS}}(\mathsf{srs}, \mathsf{x}, \boldsymbol{a}) = \mathsf{V}^{\mathcal{H}}_{\mathsf{FS}}(\mathsf{srs}, \mathsf{x}, \boldsymbol{a'}) = 1 & \boldsymbol{a} = (a_1, \ldots, a_{\mu+1}), \boldsymbol{a'} = (a'_1, \ldots, a'_{\mu+1}) \end{array}\right]$$

is upper-bounded by some negligible function $\mathsf{negl}(\lambda)$.

Note that in the above definition, srs is the SRS that $\mathcal{A}$ finalised using the update oracle UpdO, defined in Fig. 1.

Intuitively, a protocol is i-ur if it is infeasible for a PPT adversary to produce a pair of acceptable and different proofs π, π' that are the same on first i messages. We note that the definition above is also meaningful for protocols without an SRS. Intuitively in that case srs is the empty string.

3.3 Forking soundness

Note that the special soundness property (as usually defined) holds for all—even computationally unbounded—adversaries. Unfortunately, since a simulation trapdoors for P and S exist, the protocols cannot be special sound in that regard. This is because an unbounded adversary can recover the trapdoor and build a number of simulated proofs for a fake statement. Hence, we provide a weaker, yet sufficient, definition of *forking soundness*. More precisely, we state that an adversary that is able to answer correctly multiple challenges either knows the witness or can be used to break some computational assumption. However, differently from the standard definition of special soundness, we do not require from the extractor to be able to extract the witness from *any* tree of acceptable transcripts. We require that the tree be produced honestly, that is, all challenges are picked randomly — exactly as an honest verifier would pick. Intuitively, the tree is as it would be generated by a GF algorithm from the generalized forking lemma.

Definition 6 **($(\varepsilon(\lambda), k, n)$-forking soundness in the updatable setting).** *Let* $\Psi = (\mathsf{KGen}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *be an* $(2\mu + 1)$*-message proof system for a relation* $\mathbf{R}$.

For any PPT *adversary* $\mathcal{A}^{\mathsf{UpdO},\mathcal{H}}(1^\lambda; r)$ *with access to oracles* UpdO *defined in Fig. 1, and random oracle* $\mathcal{H}$*, we consider the procedure* $\mathcal{Z}$ *that provided the transcript* $(\mathsf{srs}, \mathcal{A}, r, Q_H)$ *and* $h_1, \ldots, h_q$ *runs* $\mathcal{A}$ *by providing it with random oracle queries and update oracle queries.* $\mathcal{Z}$ *returns the index* i *of the random oracle query made for challenge* k *and the proof* $\mathcal{A}$ *returns.*

Consider the algorithm $\mathsf{GF}^n_{\mathcal{Z}}$ *that rewinds* $\mathcal{Z}$ *to produce a* $(1, \ldots, n, \ldots, 1)$*-tree of transcripts.*

We say that Ψ *is* $(\varepsilon(\lambda), k, n)$*-forking if for any PPT adversary the probability that*

$$\Pr\left[\mathbf{R}(\mathsf{x}, \mathsf{w}) = 0 \,\middle|\, \begin{array}{l} r \leftarrow\$ \, \mathsf{R}(\mathcal{A}), (\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathsf{UpdO},\mathcal{H}}(1^\lambda; r), \\ (1, \mathsf{T}) \leftarrow \mathsf{GF}^m_{\mathcal{Z}}((\mathsf{srs}, \mathcal{A}, r, Q_H), Q_H), \mathsf{w} \leftarrow \mathsf{Ext}_{\mathsf{tree}}(\mathsf{T}) \end{array}\right] \leq \varepsilon(\lambda).$$

Here, srs *is the SRS that* $\mathcal{A}$ *finalised using the update oracle* UpdO*. List* $Q_{\mathcal{H}}$ *contains all* $\mathcal{A}$*'s queries to* $\mathcal{H}$ *and* $\mathcal{H}$*'s answers.*

[**Chaya:** why does GF get Q_H twice?]

Importance of the general forking lemma. To highlight the importance of the generalised forking lemma, we outline how it is used in our forking simulation-extractability proof. Let Ψ be a forking sound proof system where for an instance x the corresponding witness can be extracted from a $(1, \ldots, 1, n_k, 1, \ldots, 1)$-tree of accepting transcripts. Let $\mathcal{A}$ be the simulation-extractability adversary that outputs an accepting proof with probability at least acc. (Although we use the same acc to denote probability of $\mathcal{Z}$ outputting a non-zero i and the probability of $\mathcal{A}$ outputting an accepting proof, we claim that these probabilities are exactly the same by the way we define $\mathcal{Z}$.) Let $\mathcal{A}$ output an accepting instance-proof pair $(\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}})$; r be $\mathcal{A}$'s randomness; Q the list of simulator queries made by $\mathcal{A}$ along with Sim's answers; and $Q_{\mathcal{H}}$ be the list of all random oracle queries made by $\mathcal{A}$. All of these are given to the extractor Ext that internally runs the forking algorithm $\mathsf{GF}^{n_k}_{\mathcal{Z}}$. Algorithm $\mathcal{Z}$ takes $(\mathsf{srs}, \mathcal{A}, Q, r)$ as input y and $Q_{\mathcal{H}}$ as input $h^1_1, \ldots, h^1_q$. (For the sake of completeness, we allow $\mathsf{GF}^{n_k}_{\mathcal{Z}}$ to pick $h^1_{l+1}, \ldots, h^1_q$ responses if $Q_{\mathcal{H}}$ has only $l < q$ elements.)

Next, $\mathcal{Z}$ internally runs $\mathcal{A}(\mathsf{srs}; r)$ and responds to its random oracle and simulator queries by using $Q_{\mathcal{H}}$ and Q. Note that $\mathcal{A}$ makes the same queries as it did before it output $(\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}})$ as it is run on the same random tape and with the same answers from the simulator and random oracle. Once $\mathcal{A}$ outputs $\pi_{\mathcal{A}}$, algorithm $\mathcal{Z}$ outputs $(i, \pi_{\mathcal{A}})$, where i is the index of a random oracle query submitted by $\mathcal{A}$ to receive the challenge after the k-th message from the prover—a message where the tree of transcripts branches. Then, after the first run of $\mathcal{A}$ is done, the extractor runs $\mathcal{Z}$ again, but this time it provides fresh random oracle responses $h^2_i, \ldots, h^2_q$. Note that this is equivalent to rewinding $\mathcal{A}$ to a point just before $\mathcal{A}$ is about to ask its i-th random oracle query. The probability that the adversary produces an accepting transcript with the fresh random oracle responses is at least acc. This continues until the required number of transcripts is obtained.

We note that in the original forking lemma, the forking algorithm F, cf. **??**, gets as input only y and elements $h^1_1, \ldots, h^1_q$ are randomly picked from H internally by F. However, assuming that $h^1_1, \ldots, h^1_q$ are random oracle responses, and thus random, makes the change only notational.

We also note that the general forking lemma from Lemma 2 works for protocols with an extractor that can obtain the witness from a $(1, \ldots, 1, n_k, 1, \ldots, 1)$-tree of accepting transcripts. This limitation however does not affect the main result of this paper, i.e. showing that both Plonk and Sonic are forking simulation extractable.

4 Forking simulation-extractability—the general result

Equipped with definitional framework of Section 3, we now present the main result of this paper—a proof of forking simulation extractability of Fiat-Shamir compiled multi-round protocols.

The proof proceeds by game hopping. The games are controlled by an environment $\mathcal{E}$ that internally runs a simulation extractability adversary $\mathcal{A}$, provides it with access to a random oracle and simulator, and when necessary, rewinds it. The games differ by various breaking points, i.e. points where the environment decides to abort the game.

Denote by $\pi_{\mathcal{A}}, \pi_{\mathsf{Sim}}$ proofs returned by the adversary and the simulator respectively. We use $\pi[i]$ to denote prover's message in the i-th round of the proof (counting from 1), i.e. $(2i-1)$-th message exchanged in the protocol. $\pi[i].\mathsf{ch}$ denotes the challenge that is given to the prover after $\pi[i]$, and $\pi[i..j]$ to denote all messages of the proof including challenges between rounds i and j, but not challenge $\pi[j].\mathsf{ch}$. When it is not explicitly stated, we denote the proven instance x by $\pi[0]$ (however, there is no following challenge $\pi[0].\mathsf{ch}$).

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.

Theorem 1 (Forking simulation-extractable multi-message protocols). *Let* $\Psi = (\mathsf{KGen}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *be an interactive* $(2\mu + 1)$*-message zero-knowledge proof system for* $\mathcal{R}(1^\lambda)$*, which is trapdoor-less simulatable, has* k-ur *property with security* $\varepsilon_{\mathsf{ur}}(\lambda)$*, and is* $(\varepsilon_{\mathsf{s}}(\lambda), k, n)$*-forking sound. Let* $\mathcal{H} \colon \{0,1\}^* \to \{0,1\}^\lambda$ *be a random oracle. Then* Ψ_{FS} *is forking simulation-extractable with extraction error* $\varepsilon_{\mathsf{ur}}(\lambda)$ *against* PPT *algebraic adversaries that makes up to* q *random oracle queries and returns an acceptable proof with probability at least* acc*. The extraction probability* ext *is at least* $\mathsf{ext} \geq \frac{1}{q^{n-1}}(\mathsf{acc} - \varepsilon_{\mathsf{ur}}(\lambda))^n - \varepsilon(\lambda)$ *, for some negligible* $\varepsilon(\lambda)$*.*

Proof. **Game** G_0**:** This is the simulation-extractability game played between an adversary $\mathcal{A}$ who is given access to an oracle UpdO that defines an updatable SRS setup, a random oracle $\mathcal{H}$ and a simulation oracle SimO. There is an extractor Ext that, from a proof $\pi_{\mathcal{A}}$ for instance $\mathsf{x}_{\mathcal{A}}$ output by the adversary and from transcripts of $\mathcal{A}$'s operations is tasked to extract a witness $\mathsf{w}_{\mathcal{A}}$ such that $\mathbf{R}(\mathsf{x}_{\mathcal{A}}, \mathsf{w}_{\mathcal{A}})$ holds. $\mathcal{A}$ wins if it manages to produce an acceptable proof and the extractor fails to output a witness. In the following game hops we upper-bound the probability that this happens. Note that srs is with respect to the finalised SRS with respect to which $\mathcal{A}$ is allowed to make simulation queries.

Game G_1**:** This is identical to G_0 except that now the game is aborted if there is $(\mathsf{x}_{\mathcal{A}}, \pi_{\mathsf{Sim}}) \in Q$ such that $\pi_{\mathsf{Sim}}[1..k] = \pi_{\mathcal{A}}[1..k]$. That is, the adversary in its final proof reuses at least k messages from a simulated proof, and the proof is accepting. Denote this event by $\mathsf{Err}_{\mathsf{ur}}$.

Game 0 to Game 1: $\Pr[\mathsf{Err}_{\mathsf{ur}}] \leq \varepsilon_{\mathsf{ur}}(\lambda)$. The proof goes exactly as in **??**.**[Hamid:** since Sim-sound is not in the main body, we should write this explicitly?]

Game G_2**:** Define an adversary $\mathcal{B}$ against forking soundness such that, given access to oracles UpdO and $\mathcal{H}$, and randomness $r_{\mathcal{B}}$, it internally runs $\mathcal{A}^{\mathsf{UpdO},\mathsf{SimO},\mathcal{H}}(1^\lambda; r_{\mathcal{A}})$, where

1. $r_{\mathcal{B}}$ is split into two substrings $r_{\mathcal{A}}$ and r_{Sim};
2. $\mathcal{B}$ answers $\mathcal{A}$ update queries by asking the same query from its own update oracle. When $\mathcal{A}$ finalises an SRS srs, $\mathcal{B}$ does the same.
3. $\mathcal{B}$ answers $\mathcal{A}$'s simulator queries by programming the random oracle locally by using coins from r_{Sim}. $\mathcal{B}$ maintains a list of instance-proof pairs Q consisting of of all simulation queries made by $\mathcal{A}$, and corresponding responses.
4. Eventually when $\mathcal{A}$ outputs $(\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}})$, $\mathcal{B}$ outputs the same; $(\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}})$.

In this game, the environment aborts also when it fails to build a $(1, \ldots, 1, n, 1, \ldots, 1)$-tree of accepting transcripts T by rewinding $\mathcal{B}$. Denote that event by $\mathsf{Err}_{\mathsf{frk}}$.

Game 1 to Game 2: Note that for every accepting proof $\pi_{\mathcal{A}}$, we may assume that whenever $\mathcal{A}$ outputs a round k message $\pi_{\mathcal{A}}[k]$, then the $(\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}[1..k])$ random oracle query was made by the adversary, not the simulator[2], i.e. there is no simulated proof π_{Sim} on $\mathsf{x}_{\mathsf{Sim}}$ such that $(\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}[1..k]) = (\mathsf{x}_{\mathsf{Sim}}, \pi_{\mathsf{Sim}}[1..k])$. Otherwise, the game would be already interrupted by the error event in Game G_1. As previously, $|\Pr[\mathsf{G}_1] - \Pr[\mathsf{G}_2]| \leq \Pr[\mathsf{Err}_{\mathsf{frk}}]$.

We describe our extractor Ext here. The extractor takes as input relation $\mathbf{R}$, SRS srs, $\mathcal{B}$'s code, its randomness $r_{\mathcal{B}}$, the output instance $\mathsf{x}_{\mathcal{A}}$ and proof $\pi_{\mathcal{A}}$, and the list of random oracle queries and responses $Q_{\mathcal{H}}$. Then, Ext starts a forking algorithm $\mathsf{GF}^n_{\mathcal{Z}}(y, h_1, \ldots, h_q)$ for $y = (\mathsf{srs}, \mathcal{B}, r_{\mathcal{B}}, \mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}})$ where we set $h_1, \ldots, h_q$ to be the consecutive queries from list $Q_{\mathcal{H}}$. We run $\mathcal{B}$ internally in $\mathcal{Z}$.

To assure that in the first execution of $\mathcal{Z}$ the adversary $\mathcal{B}$ produces the same $(\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}})$ as in the extraction game, $\mathcal{Z}$ provides $\mathcal{B}$ with the same randomness $r_{\mathcal{B}}$ and answers queries to the random oracle with pre-recorded responses in $Q_{\mathcal{H}}$. Note, that since the view of the adversary when run inside $\mathcal{Z}$ is the same as its view with access to the real random oracle, it produces exactly the same output. After the first run, $\mathcal{Z}$ outputs the index i of a random oracle query that was used by $\mathcal{B}$ to compute the challenge $\pi[k].\mathsf{ch} = \mathcal{H}(\pi_{\mathcal{A}}[0..k])$ it had to answer in the $(k+1)$-th round and adversary's transcript, denoted by s_1 in GF's description. If no such query took place $\mathcal{Z}$ outputs $i = 0$.

[2] [21] calls these queries *fresh*.

Then, new random oracle responses are picked for queries indexed by $i, \dots, q$ and the adversary is rewound to the point just prior to when it gets the response to RO query $\pi_{\mathcal{A}}[0..k]$. The adversary gets a random oracle response from a new set of responses $h_i^2, \dots, h_q^2$. If the adversary requests a simulated proof after seeing h_i^2, then $\mathcal{Z}$ computes the simulated proof on its own. Eventually, $\mathcal{Z}$ outputs index i' of a query that was used by the adversary to compute $\mathcal{H}(\pi_{\mathcal{A}}[0..k])$, and a new transcript s_2. $\mathcal{Z}$ is run n times with different random oracle responses. If a tree T of n transcripts is built, then Ext internally runs the tree extractor $\mathsf{Ext}_{\mathsf{tree}}(\mathsf{T})$ and outputs what it returns.

We emphasize here the importance of the unique response property. If it does not hold then in some j-th execution of $\mathcal{Z}$ the adversary $\mathcal{A}$ (run internally in $\mathcal{B}$) could reuse a challenge that it learned from observing proofs in Q. In that case, $\mathcal{B}$ would output a proof that would make $\mathcal{Z}$ output $i = 0$, making the extractor fail. Fortunately, the case that the adversary breaks the unique response property has already been covered by the abort condition in G_1.

Denote by $\widetilde{\mathsf{acc}}$ the probability that $\mathcal{A}$ outputs a proof that is accepted and does not break k-ur-ness of Ψ. With the same probability, an accepting proof is returned by $\mathcal{B}$. Denote by $\widetilde{\mathsf{acc}}'$ the probability that algorithm $\mathcal{Z}$, defined in the general forking lemma, produces an accepting proof with a fresh challenge after round k. From the above argument, we have that $\widetilde{\mathsf{acc}} = \widetilde{\mathsf{acc}}'$.

Next, from the generalised forking lemma, cf. Lemma 2, we get that

$$\Pr[\mathsf{Err}_{\mathsf{frk}}] \leq 1 - \widetilde{\mathsf{acc}} \cdot \left(\widetilde{\mathsf{acc}}^{n-1}/q^{n-1} + (2^\lambda)!/((2^\lambda - n)! \cdot (2^\lambda)^n) - 1\right). \tag{2}$$

Game G_3**:** This game is identical to G_2 except that it aborts if $\mathsf{Ext}_{\mathsf{tree}}(\mathsf{T})$ run by Ext fails to extract a witness.

Game 2 to Game 3: Since Ψ is forking-sound the probability that $\mathsf{Ext}_{\mathsf{tree}}(\mathsf{T})$ fails is upper-bounded by $\varepsilon_{\mathsf{f}}(\lambda)$.

Since Game G_3 is aborted when it is impossible to extract a witness from T and $\mathcal{B}$ only passes proofs produced by $\mathcal{A}$, the adversary $\mathcal{A}$ cannot win. Thus, by the game-hopping argument,

$$|\Pr[\mathsf{G}_0] - \Pr[\mathsf{G}_4]| \leq 1 - \left(\frac{\widetilde{\mathsf{acc}}^n}{q^{n-1}} + \widetilde{\mathsf{acc}} \cdot \frac{(2^\lambda)!}{(2^\lambda - n)! \cdot (2^\lambda)^n} - \widetilde{\mathsf{acc}}\right) + \varepsilon_{\mathsf{ur}}(\lambda) + \varepsilon_{\mathsf{f}}(\lambda).$$

Thus the probability that extractor $\mathsf{Ext}_{\mathsf{ss}}$ succeeds is at least

$$\frac{\widetilde{\mathsf{acc}}^n}{q^{n-1}} + \widetilde{\mathsf{acc}} \cdot \frac{(2^\lambda)!}{(2^\lambda - n)! \cdot (2^\lambda)^n} - \widetilde{\mathsf{acc}} - \varepsilon_{\mathsf{ur}}(\lambda) - \varepsilon_{\mathsf{f}}(\lambda).$$

Since $\widetilde{\mathsf{acc}}$ is the probability of $\mathcal{A}$ producing an accepting transcript that does not break k-ur-ness of Ψ, then $\widetilde{\mathsf{acc}} \geq \mathsf{acc} - \varepsilon_{\mathsf{ur}}(\lambda)$, where acc is the probability of $\mathcal{A}$ outputting an accepting proof as defined in Definition 3. Thus,

$$\mathsf{ext} \geq \frac{(\mathsf{acc} - \varepsilon_{\mathsf{ur}}(\lambda))^n}{q^{n-1}} - \underbrace{(\mathsf{acc} - \varepsilon_{\mathsf{ur}}(\lambda)) \cdot \left(1 - \frac{(2^\lambda)!}{(2^\lambda - n)! \cdot (2^\lambda)^n}\right) - \varepsilon_{\mathsf{ur}}(\lambda) - \varepsilon_{\mathsf{f}}(\lambda)}_{\varepsilon(\lambda)}. \tag{3}$$

Note that the part of Eq. (3) denoted by $\varepsilon(\lambda)$ is negligible as $\varepsilon_{\mathsf{ur}}(\lambda), \varepsilon_{\mathsf{f}}(\lambda)$ are negligible, and $\frac{(2^\lambda)!}{(2^\lambda - n)! \cdot (2^\lambda)^n} \geq \left((2^\lambda - n)/2^\lambda\right)^n$ is overwhelming. Therefore,

$$\mathsf{ext} \geq q^{-(n-1)}(\mathsf{acc} - \varepsilon_{\mathsf{ur}}(\lambda))^n - \varepsilon(\lambda)\,.$$

and Ψ_{FS} is forking simulation extractable with extraction error $\varepsilon_{\mathsf{ur}}(\lambda)$. □

We conjecture that based on the recent results on state restoration soundness [**?**], which effectively allows to query the verifier multiple times on different overlapping transcripts, the q^μ loss could be avoided. However, this would reduce the class of protocols covered by our results.

5 Non-Malleability of P_{FS}

In this section, we show that P_{FS} is forking simulation-extractable. To that end, we proceed as follows. First, we show that the version of the KZG polynomial commitment scheme that is used in Plonk has the unique opening property, cf. Section 2.2 and Lemma 3. This is then used to show that P has the 2-ur property, cf. Lemma 4.

Next, we show that P is forking-sound. That is, given a number of accepting transcripts whose messages match on the first 3 rounds of the protocolm we can either extract a witness for the proven statement or use one of the transcripts to break the dlog assumption. This result is shown in the AGM, cf. Lemma 5.

Given forking-soundness and 2-ur of P, we invoke Theorem 1 and conclude that P_{FS} is forking simulation-extractable. Due to page limit, we omit description of Plonk here and refer to **??**.

5.1 Unique opening property of PC_{P}

Lemma 3. *Let PC_{P} be a batched version of a KZG polynomial commitment, cf. Fig. 3, then PC_{P} has the unique opening property (see Section 6) in the AGM with security $\varepsilon_{\mathsf{op}}(\lambda) \leq 2\varepsilon_{\mathsf{dlog}}(\lambda) + 1/\left|\mathbb{F}_p\right|$, where $\varepsilon_{\mathsf{dlog}}(\lambda)$ is security of the $(\mathsf{n}+2, 1)$-dlog assumption and $\mathbb{F}_p$ is the field used in PC_{P}.*

Proof. Let $\boldsymbol{z} = (z, z') \in \mathbb{F}_p^2$ be the two points the polynomials are evaluated at, $k \in \mathbb{N}$ be the number of the committed polynomials to be evaluated at z, $k' \in \mathbb{N}$ be the number of the committed polynomials to be evaluated at z', $\boldsymbol{c} \in \mathbb{G}^k, \boldsymbol{c'} \in \mathbb{G}^{k'}$ be the commitments, $\boldsymbol{s} \in \mathbb{F}_p^k, \boldsymbol{s'} \in \mathbb{F}_p^{k'}$ the evaluations, and $\boldsymbol{o} = (o, o') \in \mathbb{F}_p^2$ be the commitment openings. We need to show that the probability a PPT $\mathcal{A}$ opens the same commitment in two different ways is at most $\varepsilon_{\mathsf{op}}(\lambda)$, even when the commitment openings are verified in batches.

The idealised verifier checks whether the following equality, for γ, r' picked at random, holds:

$$\left(\sum_{i=1}^{k} \gamma^{i-1} \cdot \mathsf{f}_i(X) - \sum_{i=1}^{k} \gamma^{i-1} \cdot s_i\right) + r'\left(\sum_{i=1}^{k'} \gamma'^{i-1} \cdot \mathsf{f'}_i(X) - \sum_{i=1}^{k'} \gamma'^{i-1} \cdot s'_i\right) \\ \equiv \mathsf{o}(X)(X - z) + r'\mathsf{o}'(X)(X - z'). \quad (4)$$

Since r' has been picked at random from $\mathbb{F}$, probability that Eq. (4) holds while either

$$\sum_{i=1}^{k} \gamma^{i-1} \cdot \mathsf{f}_i(X) - \sum_{i=1}^{k} \gamma^{i-1} \cdot s_i \not\equiv \mathsf{o}(X)(X - z), \text{ or}$$

$$\sum_{i=1}^{k'} \gamma'^{i-1} \cdot \mathsf{f'}_i(X) - \sum_{i=1}^{k'} \gamma'^{i-1} \cdot s'_i \not\equiv \mathsf{o}'(X)(X - z')$$

is $1/\left|\mathbb{F}_p\right|$ cf. [25]. When $\sum_{i=1}^{k} \gamma^{i-1} \cdot \mathsf{f}_i(X) - \sum_{i=1}^{k} \gamma^{i-1} \cdot s_i = \mathsf{o}(X)(X - z)$ holds, polynomial $\mathsf{o}(X)$ is uniquely determined from the uniqueness of polynomial composition. Similarly, $\mathsf{o}'(X)$ is uniquely determined as well.

Any discrepancy between the idealised verifier rejection and real verifier acceptance allows one to break the discrete logarithm problem.

The reduction $\mathcal{R}_{\mathsf{dlog}}$ proceeds as follows: $\mathcal{R}_{\mathsf{dlog}}$ initializes the SRS srs using the input dlog instance, and then answers $\mathcal{A}$'s queries for SRS updates. Let srs' be the finalised SRS. Consider a proof π such that $\mathsf{ve}_{\mathsf{x},\pi}(X) \neq 0$, but $\mathsf{ve}_{\mathsf{x},\pi}(\chi') = 0$. 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 $\mathsf{ve}_{\mathsf{x},\pi}(X)$ are known. Now, $\mathcal{R}_{\mathsf{dlog}}$ computes the roots of $\mathsf{ve}_{\mathsf{x},\pi}(X)$ and finds χ' among them. Let $\chi_1, \ldots, \chi_\ell$ be the partial trapdoors of $\mathcal{A}$'s SRS updates that are extracted by $\mathcal{R}_{\mathsf{dlog}}$ from the update proofs given by $\mathcal{A}$. $\mathcal{R}_{\mathsf{dlog}}$ returns $\chi = \chi'(\chi_1\chi_2 \ldots \chi_\ell)^{-1}$.

Since any discrepancy between the idealised verifier and real verifier rejection allows one to break the discrete logarithm problem, the probability that the real verifier accepts in one of the cases above is upper-bounded by $2\varepsilon_{\mathsf{dlog}} + 1/\mathbb{F}_p$.

□

5.2 Unique response property

Lemma 4. *Let $\mathsf{PC_P}$ be commitment of knowledge with security $\varepsilon_{\mathsf{k}}(\lambda)$, $\varepsilon_{\mathsf{bind}}(\lambda)$-binding and has unique opening property with security $\varepsilon_{\mathsf{op}}(\lambda)$, then probability that a PPT adversary $\mathcal{A}$ breaks $\mathsf{P_{FS}}$'s 2-ur property is at most $\varepsilon_{\mathsf{op}} + 9 \cdot (\varepsilon_{\mathsf{bind}} + 2/\mathbb{F}_p) + \varepsilon_{\mathsf{s}} + \varepsilon_{\mathcal{H}}$, where $\varepsilon_{\mathcal{H}}$ is probability that a PPT adversary finds collision in a random oracle.*

Proof. Let $\mathcal{A}$ be an algebraic adversary tasked to break the 2-ur-ness of $\mathsf{P_{FS}}$. We show that the first 2 rounds of the protocol determines, along with the verifiers challenges, the rest of it. This is done by game hops. In the games, the adversary outputs two proofs π and π' for the same statement. 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.

Game G_0: In this game, the adversary wins if provides two transcripts that match on all 5 messages sent by the prover or finds a collision in the random oracle. Since such two transcripts cannot break the unique response property, the adversary wins this game with probability $\varepsilon_{\mathcal{H}}$ tops.

Game G_1: This game is identical to Game G_0 except that now the adversary additionally wins if it provides two transcripts that matches on the first four messages of the proof.

Game 0 to Game 1: We show that the probability that $\mathcal{A}$ wins in one game but does not in the other is negligible. Observe that in Round 5 of the proof, the adversary is given a challenge v and has to open the previously computed commitments. Since the transcripts match up to Round 4, the challenge is the same in both. Hence, to be able to give two different openings in Round 5, $\mathcal{A}$ has to break the unique opening property of the KZG commitment scheme which happens with probability $\varepsilon_{\mathsf{op}}$ tops.

Game G_2**:** This game is identical to Game G_1 except that now the adversary additionally wins if it provides two transcripts that matches on the first three messages of the proof.

Game 0 to Game 1: In Round 4 of the protocol the adversary has to provide evaluations $a_{\mathfrak{z}} = \mathsf{a}(\mathfrak{z}), b_{\mathfrak{z}} = \mathsf{b}(\mathfrak{z}), c_{\mathfrak{z}} = \mathsf{c}(\mathfrak{z}), t_{\mathfrak{z}} = \mathsf{t}(\mathfrak{z}), S_{1,\mathfrak{z}} = \mathsf{S}_{\sigma 1}(\mathfrak{z}), s_{2,\mathfrak{z}} = \mathsf{S}_{\sigma 2}(\mathfrak{z}), z_{\mathfrak{z}} = \mathsf{z}(\mathfrak{z}\omega)$ of previously committed polynomials, and compute and evaluate a linearlization polynomial r.

As before, the adversary cannot provide two different evaluations for the committed polynomials, since that would require breaking the evaluation binding property, which happens (by the union bound) with probability at most $7 \cdot (\varepsilon_{\mathsf{bind}} + 2/\left|\mathbb{F}_p\right|)$. The latter terms are since the adversary does not provide an opening for each of the commitment separately, but only in a batched way. That comes with $1/\mathbb{F}_p$ of security loss. Another $1/\mathbb{F}_p$ security loss comes from the fact that the verification of commitment openings are batched as well.

The adversary cannot also provide two different linearization polynomials r^0 and r^1 evaluations $r^0_{\mathfrak{z}}$ and $r^1_{\mathfrak{z}}$ as the linearization polynomial is determined by values known to the verifier who also can compute a commitment to $\mathsf{r}(X)$ equal $[\mathsf{r}(\chi)]_1$ by its own. The evaluation of r provided by the adversary is later checked, as $\mathcal{A}$ opens the commitment in Round 5. Hence, the probability that the adversary manages to build two convincing proofs that differ in evaluations $r_{\mathfrak{z}}$ and $r'_{\mathfrak{z}}$ is at most $\varepsilon_{\mathsf{bind}} + 2/\left|\mathbb{F}_p\right|$.

Hence, the probability that adversary wins in one game but does not in the other is upper-bounded by $8 \cdot (\varepsilon_{\mathsf{bind}} + 2/\mathbb{F}_p)$

Game G_3**:** This game is identical to Game G_2 except that now the adversary additionally wins if it provides two transcripts that matches on the first two messages of the proof.

Game 2 to Game 3: In Round 3 the adversary computes the quotient polynomial $\mathsf{t}(X)$ and provides its commitment that compounds of three separate commitments $[\mathsf{t}_{\mathsf{lo}}(\chi), \mathsf{t}_{\mathsf{mid}}(\chi), \mathsf{t}_{\mathsf{hi}}(\chi)]_1$. Let $\left[\mathsf{t}^0_{\mathsf{lo}}(\chi), \mathsf{t}^0_{\mathsf{mid}}(\chi), \mathsf{t}^0_{\mathsf{hi}}(\chi)\right]_1$ be the commitments output by the adversary in one transcript, and $\left[\mathsf{t}^1_{\mathsf{lo}}(\chi), \mathsf{t}^1_{\mathsf{mid}}(\chi), \mathsf{t}^1_{\mathsf{hi}}(\chi)\right]_1$ the commitments provided in the other. Since the commitment scheme is deterministic, the adversary cannot come up with two different valid commitments for the same polynomial.

If the adversary picks two different polynomials: $\mathsf{t}^0(X)$, that is committed as $\left[\mathsf{t}^0_{\mathsf{lo}}(\chi), \mathsf{t}^0_{\mathsf{mid}}(\chi), \mathsf{t}^0_{\mathsf{hi}}(\chi)\right]_1$, and $\mathsf{t}^1(X)$ that is committed as $\left[\mathsf{t}^1_{\mathsf{lo}}(\chi), \mathsf{t}^1_{\mathsf{mid}}(\chi), \mathsf{t}^1_{\mathsf{hi}}(\chi)\right]_1$, then one of them has to be computed incorrectly.

Importantly, polynomial $\mathsf{t}(X)$ assures that the constraints of the system hold. Hence, the probability that one of $\mathsf{t}^0(X)$, $\mathsf{t}^1(X)$ is computed incorrectly, the adversary gives and opens acceptably a commitment to it, and the proof is acceptable, is upper bounded by the soundness of the proof system ε_{s}. Alternatively, $\mathcal{A}$ may compute a commitment to an invalid $\mathsf{t}^0(X)$ (or $\mathsf{t}^1(X)$) and later open the commitment at $\mathfrak{z}$ to $\mathsf{t}(\mathfrak{z})$. That is, give an evaluation from the correct polynomial $\mathsf{t}(X)$. Since the commitment scheme is evaluation binding, probability of such event is upper bounded by $\varepsilon_{\mathsf{bind}} + 2/\left|\mathbb{F}_p\right|$.

Conclusion: Taking all the games together, probability that $\mathcal{A}$ wins in G_3 is upper-bounded by

$$2 \cdot \varepsilon_{\mathsf{op}} + 9 \cdot (\varepsilon_{\mathsf{bind}} + 2/\mathbb{F}_p) + \varepsilon_{\mathcal{H}} + \varepsilon_{\mathsf{s}}.$$

□

5.3 Forking soundness

Lemma 5. *Let KZG be hiding with security $\varepsilon_{\mathsf{hid}}(\lambda)$, P's idealized verifier fail with probability $\varepsilon_{\mathsf{id}}(\lambda)$, and $(\mathsf{n} + 2, 1)$-dlog problem be $\varepsilon_{\mathsf{dlog}}(\lambda)$ hard. Then P is $(\varepsilon_{\mathsf{id}}(\lambda) + \varepsilon_{\mathsf{dlog}}(\lambda) + 8 \cdot S \cdot \varepsilon_{\mathsf{hid}}(\lambda) +, 3, 3\mathsf{n} + 1)$-forking sound against algebraic adversary $\mathcal{A}$ who makes up to $S = \mathsf{poly}(\lambda)$ simulation oracle queries.[**Hamid:** In the new definition, $\mathcal{A}$ does not have access to the simulation oracle; so this should be changed!]*

Proof. The main idea of the proof is to show that an adversary who breaks forking soundness can be used to break hiding properties of the polynomial commitment scheme or a dlog problem instance. The proof goes by game hops. Let T be the tree produced by $\mathcal{T}$ by rewinding $\mathcal{A}$. Note that since the tree branches after Round 3, 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. The tree branches after the third round of the protocol 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 all the transcripts it produced are acceptable by the ideal verifier, i.e. $\mathsf{ve}_{\mathsf{x},\pi}(X) = 0$, cf. **??**, and none of commitments $[\mathsf{a}(\chi), \mathsf{b}(\chi), \mathsf{c}(\chi), \mathsf{z}(\chi), \mathsf{t}_{\mathsf{lo}}(\chi), \mathsf{t}_{\mathsf{mid}}(\chi), \mathsf{t}_{\mathsf{hi}}(\chi)]_1$ use elements from a simulated proof, and the extractor fails to extract a valid witness out of the proof.

Probability that $\mathcal{A}$ wins Game 0 is negligible: Probability of $\mathcal{A}$ winning this game is $\varepsilon_{\mathsf{id}}(\lambda)$ as the protocol P, instantiated with the idealised verification equation, is perfectly knowledge sound except with negligible probability of the idealised verifier failure $\varepsilon_{\mathsf{id}}(\lambda)$. Hence for a valid proof π for a statement x there exists a witness w, such that $\mathbf{R}(\mathsf{x}, \mathsf{w})$ holds. Note that since the $\mathcal{T}$ produces $(3\mathsf{n} + 1)$ acceptable 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 the transcripts are acceptable by an idealised verifier, the equality between polynomial $\mathsf{t}(X)$ and combination of polynomials $\mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X), \mathsf{z}(X)$ described in Round 3 of the protocol holds. Hence, $\mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X)$ encodes the valid witness for the proven statement. Since $\mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X)$ are of degree at most $(\mathsf{n} + 2)$ and there is more than $(\mathsf{n} + 2)$ their evaluations known, $\mathsf{Ext}_{\mathsf{tree}}$ can recreate polynomials' coefficients by interpolation and reveal the witness with probability 1. Hence, the probability that extraction fails in that case is upper-bounded by probability of an idealised verifier failing $\varepsilon_{\mathsf{id}}(\lambda)$, which is negligible.

Game 1: In this game the adversary additionally wins if it produces a transcript in T such that $\mathsf{ve}_{\mathsf{x},\pi}(\chi) = 0$, but $\mathsf{ve}_{\mathsf{x},\pi}(X) \neq 0$, and none of commitments $[\mathsf{a}(\chi), \mathsf{b}(\chi), \mathsf{c}(\chi), \mathsf{z}(\chi), \mathsf{t}_{\mathsf{lo}}(\chi), \mathsf{t}_{\mathsf{mid}}(\chi), \mathsf{t}_{\mathsf{hi}}(\chi)]_1$ use elements from a simulated proof. The first condition means that the ideal verifier does not accept the proof, but the real verifier does.

Game 0 to Game 1: Assume the adversary wins in Game 1, but does not win in Game 0. We show that such adversary may be used to break the dlog assumption. More precisely, let $\mathcal{T}$ be an algorithm that for relation $\mathbf{R}$ and randomly picked $\mathsf{srs} \leftarrow_{\$} \mathsf{KGen}(\mathbf{R})$ produces a tree of acceptable transcripts such that the winning condition of the game holds. Let $\mathcal{R}_{\mathsf{dlog}}$ be a reduction that gets as input an $(\mathsf{n}+2, 1)$-dlog instance $[1, \ldots, \chi^{\mathsf{n}}]_1, [\chi]_2$ [**Hamid:** shouldn't be $\left[1, \ldots, \chi^{\mathsf{n}+2}\right]_1, [\chi]_2$] and is tasked to output χ.

The reduction $\mathcal{R}_{\mathsf{dlog}}$ proceeds as follows. [**Hamid:**

1. Build a SRS srs using the input dlog instance. Answer $\mathcal{A}$'s queries for SRS updates and set the honest update of the SRS to be srs. Let srs' be the finalised SRS. Start $\mathcal{T}(\mathcal{A}, \mathsf{srs}')$;
2. 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.
3. Find $\mathsf{ve}_{\mathsf{x},\pi}(X)$ zero points and find χ' among them.
4. Let $\chi_1, \ldots, \chi_\ell$ be the partial trapdoors of $\mathcal{A}$'s SRS updates. These trapdoors can be extracted by the reduction from the update proofs given by $\mathcal{A}$.
5. Return $\chi = \chi'(\chi_1 \chi_2 \ldots \chi_\ell)^{-1}$.

] Hence, the probability that the adversary wins Game 1 is upper-bounded by $\varepsilon_{\mathsf{dlog}}(\lambda)$.

Game 2: In this game the adversary additionally wins if at least one of the commitments $\mathsf{a}(\chi), \mathsf{b}(\chi), \mathsf{c}(\chi), \mathsf{z}(\chi)$ utilizes a commitment that comes from a simulated proof; for example, $\mathcal{A}$ could compute its commitment to $\mathsf{c}(X)$ as follows: it picks a polynomial $\mathsf{p}(X)$, computes $[\mathsf{p}(\chi)]_1$, and outputs commitment $[\mathsf{c}(\chi)]_1 = [\mathsf{p}(\chi)]_1 + c$, where c is a commitment output by a simulator. In the following, w.l.o.g, we assume that $\mathcal{A}$ uses some simulated element to compute commitment $[\mathsf{c}(\chi)]_1$.

Game 1 to Game 2: Given adversary $\mathcal{A}$ that wins in Game 2, but not in Game 1, we show a reduction $\mathcal{R}$ that uses $\mathcal{A}$ and $\mathcal{T}$ to break hiding property of the commitment scheme. $\mathcal{R}$ proceeds as follows:

1. [**Hamid:** Given polynomial commitment SRS $\mathsf{srs}_{\mathsf{PC}}$, produce Plonk's SRS srs.]
2. Pick random polynomials $\mathsf{p}(X), \mathsf{p}'(X) \in \mathbb{F}^{<|\mathsf{H}|}[X]$, hiding parameter $k = 2$ and send them to the polynomial commitment challenger C.
3. From the challenger get the challenge commitment c.
4. Let S be the upper bound on the number of simulator oracles queries the adversary can make.
5. Guess which simulator's response is going to be used by $\mathcal{A}$ in its proof. Let s be the index of this response.[**Hamid:** should be changed!]
6. Guess which of the simulated polynomials in response s will be used. Let i be the index of this polynomial.[**Hamid:** should be changed!]
7. Let $\mathcal{T}'$ be an algorithm that behaves exactly as $\mathcal{T}$, except when $\mathcal{A}$ asks for s-th simulated proof, $\mathcal{T}'$'s internal procedure $\mathcal{B}'$ provides $\mathcal{A}$ with a simulated proof such that instead of randomly picked commitment $\mathsf{c}(\chi)$ it gives c. Michal 9.9: Alternatively, we can parametrize $\mathcal{T}$ by $\mathcal{B}$.
8. Start $\mathcal{T}'(\mathcal{A}, \mathsf{srs})$ and get the tree T.
9. Since T contains $\mathsf{n}+1$ evaluations of $\mathsf{c}(X)$, the polynomial can be reconstructed.

10. Since $\mathcal{A}$ is algebraic, $\mathcal{R}$ learns composition of $\mathsf{c}(X)$ in the srs and simulated elements.
11. Hence $\mathcal{R}$ learns whether c is a commitment to $\mathsf{p}(X)$ or $\mathsf{p}'(X)$.
12. $\mathcal{R}$ returns its guessing bit to C.

Game 3: In this game the adversary additionally wins if at least one of the commitments $\mathsf{t}_{\mathsf{lo}}(\chi), \mathsf{t}_{\mathsf{mid}}(\chi), \mathsf{t}_{\mathsf{hi}}(\chi)$ comes from a simulated proof.

Game 2 to Game 3: Given adversary $\mathcal{A}$ that wins in Game 3, but not in Game 2, we show a reduction $\mathcal{R}$ that uses $\mathcal{A}$ and $\mathcal{T}$ to break hiding property of the commitment scheme. $\mathcal{R}$ proceeds as follows:

1. Guess the simulation query index s the polynomial(s) come from and whether the polynomial is $\mathsf{t}_{\mathsf{lo}}(X), \mathsf{t}_{\mathsf{mid}}(X)$, or $\mathsf{t}_{\mathsf{hi}}(X)$. Denote by $i \in [1\,..\,3]$ the index of the guessed polynomial. W.l.o.g. assume $i = 1$, i.e. it is $\mathsf{t}_{\mathsf{lo}}(X)$.
2. Produce two random polynomials $\mathsf{p}_0(X)$ and $\mathsf{p}_1(X)$ and send them to the challenger C. Get commitment c.
3. Let $\mathcal{T}'$ be an algorithm that behaves exactly as $\mathcal{T}$, except when $\mathcal{A}$ asks for s-th simulated proof, $\mathcal{T}'$'s internal procedure $\mathcal{B}'$ provides $\mathcal{A}$ with a simulated proof such that:
   (a) Start making the simulated proof as a trapdoor-less simulator would.
   (b) Before polynomials $\mathsf{t}_{\mathsf{lo}}(X), \mathsf{t}_{\mathsf{mid}}, \mathsf{t}_{\mathsf{hi}}$ are computed, pick random $\mathfrak{z}$ and get evaluation $p_{\mathfrak{z}} = \mathsf{p}_{\mathsf{b}}(\mathfrak{z})$, i.e. evaluate the polynomial in c at $\mathfrak{z}$.
   (c) Let $\widetilde{\mathsf{t}_{\mathsf{lo}}}$ be the evaluation of the simulated $\mathsf{t}_{\mathsf{lo}}(\mathfrak{z})$.
   (d) Pick r such that $p_{\mathfrak{z}} + r = \mathsf{t}_{\mathsf{lo}}(\mathfrak{z})$.
   (e) For the commitment of t_{lo} output $c' = c + [r]_1$.
   (f) Compute the rest of the simulated proof as a simulator would.
4. Let $\mathcal{T}'$ be an algorithm that behaves exactly as $\mathcal{T}$, except when $\mathcal{A}$ asks for s-th simulated proof, $\mathcal{T}'$'s internal procedure $\mathcal{B}'$ provides $\mathcal{A}$ with a simulated proof such that instead of a simulated $\mathsf{t}_{\mathsf{lo}}(\chi)$ it gives c'.
5. Start $\mathcal{T}'(\mathcal{A}, \mathsf{srs})$ and get the tree T.
6. Since T contains $\mathsf{n} + 1$ evaluations of $\mathsf{t}_{\mathsf{lo}}(X)$, the polynomial can be reconstructed.
7. Since $\mathcal{A}$ is algebraic, $\mathcal{R}$ learns composition of $\mathsf{t}_{\mathsf{lo}}(X)$ in the srs and simulated elements.
8. Hence $\mathcal{R}$ learns whether c is a commitment to p or p'.
9. $\mathcal{R}$ returns its guessing bit to C.

5.4 Honest verifier zero-knowledge

Lemma 6. *Let* P *be zero knowledge with security* $\varepsilon_{\mathsf{zk}}(\lambda)$. *Let* $(\mathsf{R}, \mathsf{S}, \mathsf{T}, \mathsf{f}, 1)$*-uber assumption for* $\mathsf{R}, \mathsf{S}, \mathsf{T}, \mathsf{f}$ *as defined in Eq.* (5) *hold with security* $\varepsilon_{\mathsf{uber}}(\lambda)$. *Then* P *is computationally honest verifier zero-knowledge with simulator* Sim *that does not require a SRS trapdoor with security* $\varepsilon_{\mathsf{zk}}(\lambda) + \varepsilon_{\mathsf{uber}}(\lambda)$.[3]

[3] The simulator works as a simulator for proofs that are zero-knowledge in the standard model. However, we do not say that Plonk is HVZK in the standard model as proof of that *requires* the SRS simulator.

Proof. The proof goes by game-hopping. The environment that controls the games provides the adversary with a SRS srs, then the adversary outputs an instance–witness pair (x, w) and, depending on the game, is provided with either real or simulated proof for it. In the end of the game the adversary outputs either 0 if it believes that the proof it saw was provided by the simulator and 1 in the other case.

Game G_0: In this game $\mathcal{A}(\mathsf{srs})$ picks an instance–witness pair (x, w) and gets a real proof π for it.

Game G_1: In this game for $\mathcal{A}(\mathsf{srs})$ picks an instance–witness pair (x, w) and gets a proof π that is simulated by a simulator Sim_χ which utilises for the simulation the SRS trapdoor and proceeds as described in **??**.

Game 0 to Game 1: Since Plonk is zero-knowledge, probability that $\mathcal{A}$ outputs a different bit in both games is negligible. Hence $\left|\Pr[\mathsf{G}_0] - \Pr[\mathsf{G}_1]\right| \leq \varepsilon_{\mathsf{zk}}(\lambda)$.

Game G_2: In this game $\mathcal{A}(\mathsf{srs})$ picks an instance–witness pair (x, w) and gets a proof π simulated by the simulator Sim which proceeds as follows.

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

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

In Round 3 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* Round 3. Then Sim computes evaluations $\mathsf{a}(\mathfrak{z}), \mathsf{b}(\mathfrak{z}), \mathsf{c}(\mathfrak{z}), \mathsf{S}_{\sigma 1}(\mathfrak{z}), \mathsf{S}_{\sigma 2}(\mathfrak{z}), \mathsf{PI}(\mathfrak{z}), \mathsf{L}_1(\mathfrak{z}), \mathsf{Z}_\mathsf{H}(\mathfrak{z}), \mathsf{z}(\mathfrak{z}\omega)$ and computes $\mathsf{t}(X)$ honestly. Since for a random $\mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X), \mathsf{z}(X)$ the constraint system is (with overwhelming probability) not satisfied and the constraints-related polynomials are not divisible by $\mathsf{Z}_\mathsf{H}(X)$, hence $\mathsf{t}(X)$ is a rational function rather than a polynomial. Then, the simulator evaluates $\mathsf{t}(X)$ at $\mathfrak{z}$ and picks randomly a degree-$(3\mathsf{n}-1)$ polynomial $\tilde{\mathsf{t}}(X)$ such that $\mathsf{t}(\mathfrak{z}) = \tilde{\mathsf{t}}(\mathfrak{z})$ and publishes a commitment $\left[\tilde{\mathsf{t}}_{\mathsf{lo}}(\chi), \tilde{\mathsf{t}}_{\mathsf{mid}}(\chi), \tilde{\mathsf{t}}_{\mathsf{hi}}(\chi)\right]_1$. After this round the simulator outputs $\mathfrak{z}$ as a challenge.

In the next round, the simulator computes polynomial $\mathsf{r}(X)$ as an honest prover would, cf. **??** and evaluates $\mathsf{r}(X)$ at $\mathfrak{z}$.

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

Game 1 to Game 2: We now describe the reduction $\mathcal{R}$ which relies on the $(\mathsf{R}, \mathsf{S}, \mathsf{T}, \mathsf{F}, 1)$-uber assumption, cf. **??** where $\mathsf{R}, \mathsf{S}, \mathsf{T}, \mathsf{F}$ are polynomials over variables $\boldsymbol{B} = B_1, \ldots, B_9$ and are defined as follows. Let $E = \{\{2\}, \{3, 4\}, \{5, 6\}, \{7, 8, 9\}\}$ and $E' = E \setminus \{2\}$. Let

$$
\begin{aligned}
\mathsf{F}(\boldsymbol{B}) = {} & \{B_1\} \cup \{B_1 B_i \mid i \in A,\ A \in E'\} \cup \{B_1 B_i B_j \mid i \in A, j \in B,\ A, B \in E', B \neq A\} \cup \\
& \{B_1 B_i B_j B_k \mid i \in A, j \in B, k \in C,\ A, B, C \in E', A \neq B \neq C \neq A\}\,, \\
\mathsf{R}(\boldsymbol{B}) = {} & \{B_i \mid i \in A,\ A \in E\} \cup \{B_i B_j \mid i \in A, j \in B,\ A \neq B, A, B \in E\} \cup \qquad (5)
\end{aligned}
$$

$$\begin{aligned}
&\{B_i B_j B_k \mid i \in A,\ j \in B,\ k \in C,\ A, B, C \text{ all different and in } E\} \cup \\
&\{B_i B_j B_k B_l \mid i \in A,\ j \in B,\ k \in C,\ l \in D,\ A, B, C, D \text{ all different and in } E\} \\
&\setminus \mathsf{F}(\boldsymbol{B}), \\
\mathsf{S}(\boldsymbol{B}) = \emptyset, \qquad & \mathsf{T}(\boldsymbol{B}) = \emptyset.
\end{aligned} \tag{6}$$

That is, the elements of R are all singletons, pairs, triplets and quadruplets of B_i variables that occur in polynomial $\mathsf{t}(\boldsymbol{B})$ except the challenge element $\mathsf{f}(\boldsymbol{B})$ which are all elements that depends on a variable B_1. Variables $\boldsymbol{B}$ are evaluated to randomly picked $\boldsymbol{b} = b_1, \ldots, b_9$.

The reduction $\mathcal{R}$ learns $[\mathsf{R}]_1$ and challenge $[\boldsymbol{w}]_1 = [w_1, \ldots, w_{12}]_1$ where $\boldsymbol{w}$ is either a vector of evaluations $\mathsf{F}(\boldsymbol{b})$ or a sequence of random values $y_1, \ldots, y_{12}$, for the sake of concreteness we state $w_1 = b_1$ or $w_1 = y_1$ (depending on the chosen random bit). Then it picks χ, $\mathfrak{z}$ and computes the SRS srs from χ. Elements b_i are interpreted as polynomials in X that are evaluated at χ, i.e. $b_i = b_i(\chi)$. Next, $\mathcal{R}$ sets for $\xi_i, \zeta_i \leftarrow\!\!\$\, \mathbb{F}_p$ $\left[\tilde{\mathsf{b}}_1(X)\right]_1 = (X - \mathfrak{z})(X - \omega\mathfrak{z}) [w_1]_1 (X) + \xi_i (X - \mathfrak{z}) [1]_1 + \zeta_i (X - \omega\mathfrak{z}) [1]_1$,, and $\left[\tilde{\mathsf{b}}_i(X)\right]_1 = (X - \mathfrak{z})(X - \omega\mathfrak{z}) [b_i]_1 (X) + \xi_i (X - \mathfrak{z}) [1]_1 + \zeta_i (X - \omega\mathfrak{z}) [1]_1$, for $i \in [2\,..\,9]$.

Denote by $\tilde{b}_i$ evaluations of $\tilde{\mathsf{b}}_i$ at χ. The reduction computes all $\left[\tilde{b}_i \tilde{b}_j\right]_1, \left[\tilde{b}_i \tilde{b}_j \tilde{b}_k\right]_1, \left[\tilde{b}_i \tilde{b}_j \tilde{b}_k \tilde{b}_l\right]_1$ such that $\left[B_i B_j, B_i B_j B_k, B_i B_j B_k B_l\right]_1 \in \mathsf{R}$. This is possible since $\mathcal{R}$ knows all singletons $[w_1, b_2, \ldots, b_9]_1$ and pairs $\left[b_i b_j\right]_1 \in \mathsf{R}$ which can be used to compute all required pairs $\left[\tilde{b}_i \tilde{b}_j\right]_1$:

$$\begin{aligned}
\left[\tilde{b}_i \tilde{b}_j\right]_1 = {} & ((\chi - \mathfrak{z})(\chi - \omega\mathfrak{z}) [b_i]_1 + \xi_i (\chi - \mathfrak{z}) [1]_1 + \zeta_i (\chi - \omega\mathfrak{z}) [1]_1) \cdot \\
& ((\chi - \mathfrak{z})(\chi - \omega\mathfrak{z}) \left[b_j\right]_1 + \xi_j (\chi - \mathfrak{z}) [1]_1 + \zeta_j (\chi - \omega\mathfrak{z}) [1]_1) = \\
& ((\chi - \mathfrak{z})(\chi - \omega\mathfrak{z}))^2 \left[b_i b_j\right]_1 + ((\chi - \mathfrak{z})(\chi - \omega\mathfrak{z}) [b_i]_1 (\xi_j (\chi - \mathfrak{z}) [1]_1 + \zeta_j (\chi - \omega\mathfrak{z}) [1]_1) + \\
& ((\chi - \mathfrak{z})(\chi - \omega\mathfrak{z}) \left[b_j\right]_1 (\xi_i (\chi - \mathfrak{z}) [1]_1 + \zeta_i (\chi - \omega\mathfrak{z}) [1]_1) + \psi,
\end{aligned}$$

where ψ compounds of $\xi_i, \xi_j, \zeta_i, \zeta_j, \mathfrak{z}, \omega\mathfrak{z}, \chi$ which are all known by $\mathcal{R}$ and no b_i nor b_j. Analogously for the triplets and quadruplets and elements dependent on $\boldsymbol{w}$.

Next the reduction runs the adversary $\mathcal{A}(\mathsf{srs})$ and obtains from $\mathcal{A}$ an instance–witness pair (x, w). $\mathcal{R}$ now prepares a simulated proof as follows:

Round 1 $\mathcal{R}$ computes $[\mathsf{a}(\chi)]_1$ using as randomisers $\left[\tilde{b}_1\right]_1, \left[\tilde{b}_2\right]_1$ and setting $\mathsf{w}_i = 0$, for $i \in [1\,..\,3\mathsf{n}]$. Similarly it computes $[\mathsf{b}(\chi)]_1, [\mathsf{c}(\chi)]_1$. $\mathcal{R}$ publishes the obtained values and picks a Round 1 challenge β, γ at random. Note that regardless $w_1 = b_1$ or a random element, $[a(\chi)]_1$ is random. Thus $\mathcal{R}$'s output has the same distribution as output of a real prover.

Round 2 $\mathcal{R}$ computes $[\mathsf{z}(\chi)]_1$ using $\tilde{b}_7, \tilde{b}_8, \tilde{b}_9$ and publishes it. Then it picks randomly the challenge α. This round output is independent on b_1 thus $\mathcal{R}$'s output is indistinguishable from the prover's.

Round 3 The reduction computes $\mathsf{t}_{\mathsf{lo}}(\chi), \mathsf{t}_{\mathsf{mid}}(\chi), \mathsf{t}_{\mathsf{hi}}(\chi)$, which all depend on b_1. To that end $\left[\tilde{b}_1\right]_1$ is used. Note that if $\boldsymbol{w}$ is a vector of $\mathsf{F}(b_1, \ldots, b_9)$ evaluations then $[\mathsf{t}_{\mathsf{lo}}(\chi), \mathsf{t}_{\mathsf{mid}}(\chi), \mathsf{t}_{\mathsf{hi}}(\chi)]_1$ is the same as the real prover's. Alternatively, if $\boldsymbol{w}$ is a vector of random values, then $\mathsf{t}_{\mathsf{lo}}(\chi), \mathsf{t}_{\mathsf{mid}}(\chi), \mathsf{t}_{\mathsf{hi}}(\chi)$ are all random polynomials which evaluates at $\mathfrak{z}$ to the same value as the polynomials computed by the real prover. That is, in that case $\mathsf{t}_{\mathsf{lo}}(\chi), \mathsf{t}_{\mathsf{mid}}(\chi), \mathsf{t}_{\mathsf{hi}}(\chi)$ are as the simulator Sim would compute. Eventually, $\mathcal{R}$ outputs $\mathfrak{z}$.

Round 4 The reduction outputs $\mathsf{a}(\mathfrak{z}), \mathsf{b}(\mathfrak{z}), \mathsf{c}(\mathfrak{z}), \mathsf{S}_{\sigma 1}(\mathfrak{z}), \mathsf{S}_{\sigma 2}(\mathfrak{z}), \mathsf{t}(\mathfrak{z}), \mathsf{z}(\omega\mathfrak{z})$. For the sake of concreteness, denote by $S = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{t}, \mathsf{z}\}$. Although for a polynomial $\mathsf{p} \in S$, reduction $\mathcal{R}$ does not know $\mathsf{p}(\chi)$ or even do not know all the coefficients of p, the polynomials in S was computed such that the reduction always knows their evaluation at $\mathfrak{z}$ and $\omega\mathfrak{z}$.

Round 5 $\mathcal{R}$ computes the openings of the polynomial commitments assuring that evaluations at $\mathfrak{z}$ it provided were computed honestly.

If the adversary $\mathcal{A}$'s output distribution differ in Game G_1 and G_2 then the reduction uses it to distinguish between $\boldsymbol{w} = \mathsf{F}(b_1, \ldots, b_9)$ and $\boldsymbol{w}$ being random, thus $|\Pr[\mathsf{G}_1] - \Pr[\mathsf{G}_2]| \leq \varepsilon_{\mathsf{uber}}(\lambda)$. Eventually, $|\Pr[\mathsf{G}_0] - \Pr[\mathsf{G}_2]| \leq \varepsilon_{\mathsf{zk}}(\lambda) + \varepsilon_{\mathsf{uber}}(\lambda)$. □

5.5 Simulation extractability of P_{FS}

Since Lemmas 4 and 5 hold, P is 2-ur and forking sound. We now make use of Theorem 1 and show that P_{FS} is forking simulation-extractable as defined in Definition 3.

Corollary 1 (Forking simulation extractability of P_{FS}). *Assume an idealised* P *verifier fails at most with probability $\varepsilon_{\mathsf{id}}(\lambda)$, the discrete logarithm advantage is bounded by $\varepsilon_{\mathsf{dlog}}(\lambda)$ and the* PC_{P} *is a commitment of knowledge with security $\varepsilon_{\mathsf{k}}(\lambda)$, binding security $\varepsilon_{\mathsf{bind}}(\lambda)$ and has unique opening property with security $\varepsilon_{\mathsf{op}}(\lambda)$. Let $\mathcal{H}\colon \{0,1\}^* \rightarrow \{0,1\}^{\lambda}$ be a random oracle. Let $\mathcal{A}$ be an algebraic adversary that can make up to q random oracle queries, up to S simulation oracle queries, and outputs an acceptable proof for* P_{FS} *with probability at least* acc. *Then* P_{FS} *is forking simulation-extractable with extraction error $\eta = \varepsilon_{\mathsf{ur}}(\lambda)$. The extraction probability* ext *is at least*

$$\mathsf{ext} \geq \frac{1}{q^{3(\varepsilon_{\mathsf{id}}(\lambda)+\varepsilon_{\mathsf{dlog}}(\lambda))}}(\mathsf{acc} - \varepsilon_{\mathsf{k}}(\lambda) - 2 \cdot \varepsilon_{\mathsf{bind}}(\lambda) - \varepsilon_{\mathsf{op}}(\lambda))^{3\mathsf{n}+1} - \varepsilon(\lambda)\,,$$

for some negligible $\varepsilon(\lambda)$ and n *being the number of constraints in the proven circuit.*

6 Polynomial Commitment Schemes

Figs. 3 and 4 present variants of KZG polynomial commitment schemes used in Plonk and Sonic. The key generation algorithm KGen takes as input a security parameter 1^{λ} and a parameter max which determines the maximal degree of the committed polynomial. We assume that max can be read from the output SRS. While the figures only describe trusted SRS setup, it is not hard to lift the SRS generation into the updatable setting by defining the extra algorithms Upd, $\mathsf{VerifySRS}$ as defined in Section 2.5.

The following properties are expected of a secure polynomial commitment PC. Note that since we are in the updatable setting, srs in the following definitions is the SRS that $\mathcal{A}$ finalises 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 $\mathcal{A}$

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

We say that PC has the unique opening property if the following holds:

Opening uniqueness: 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, $\boldsymbol{s} \in \mathbb{F}_p^k$ the evaluations, and $\boldsymbol{o} \in \mathbb{F}_p^l$ be the commitment openings. Then for every PPT adversary $\mathcal{A}$

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

Intuitively, opening uniqueness assures that there is only one valid opening for the committed polynomial and given evaluation point. This property is crucial in showing forking simulation-extractability of Plonk and Sonic. We show that the Plonk's and Sonic's polynomial commitment schemes satisfy this requirement in Lemma 3 and **??** respectively.

We also formalize notion of k-hiding property of a polynomial commitment scheme

Hiding: 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

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

Here, O_C is a challenge oracle that

1. takes polynomials f_0, f_1 provided by the adversary and parameter k,
2. samples bit b,
3. samples vector $\boldsymbol{a} \in \mathbb{F}^k$,
4. computes polynomial, $f_b'(X) = f_b + \mathsf{Z_H}(X)(a_0 + a_1 X + \ldots a_{k-1} X^{k-1})$,
5. outputs polynomial commitment $c = f_b'(\chi)$,
6. 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 For every PPT adversary $\mathcal{A}$ who produces commitment c, evaluation s and opening o there exists a PPT extractor Ext such that

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

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

Intuitively when a commitment scheme is “of knowledge” then if an adversary produces a (valid) commitment c, which it can open, then it also knows the underlying polynomial f which commits to that value. [42] shows, using AGM, that $\mathsf{PC_S}$ is a commitment of knowledge. The same reasoning could be used to show that property for $\mathsf{PC_P}$.

```
$\mathsf{KGen}(1^\lambda, \mathsf{max})$
$\chi \leftarrow\$ \mathbb{F}_p^2$
return $\left[1, \ldots, \chi^{\mathsf{n}+2}\right]_1, [\chi]_2$

$\mathsf{Com}(\mathsf{srs}, \mathsf{f}(X))$
return $[\boldsymbol{c}]_1 = [\mathsf{f}(\chi)]_1$

$\mathsf{Op}(\mathsf{srs}, \boldsymbol{\gamma}, \boldsymbol{z}, \boldsymbol{s}, \mathsf{f}(X))$
for $i \in [1 \mathinner{..} |\boldsymbol{z}|]$ do
    $\mathsf{o}_i(X) \leftarrow \sum_{j=1}^{t_i} \gamma_i^{j-1} \frac{\mathsf{f}_{i,j}(X) - \mathsf{f}_{i,j}(z_i)}{X - z_i}$
return $\boldsymbol{o} = [\mathsf{o}(\chi)]_1$

$\mathsf{Vf}(\mathsf{srs}, [c]_1, \boldsymbol{z}, \boldsymbol{s}, [\mathsf{o}(\chi)]_1)$
$\boldsymbol{r} \leftarrow \mathbb{F}_p^{|\boldsymbol{z}|}$
for $i \in [1 \mathinner{..} |\boldsymbol{z}|]$ do
    if $\sum_{i=1}^{|\boldsymbol{z}|} r_i \cdot \left[\sum_{j=1}^{t_j} \gamma_i^{j-1} c_{i,j} - \sum j = 1^{t_j} s_{i,j}\right]_1 \bullet [1]_2 +$
    $\sum_{i=1}^{|\boldsymbol{z}|} r_i z_i o_i \bullet [1]_2 \neq \left[-\sum_{i=1}^{|\boldsymbol{z}|} r_i o_i\right]_1 \bullet [\chi]_2$ then
    return $0$
return $1$.
```

Fig. 3: $\mathsf{PC_P}$ polynomial commitment scheme.

```
$\mathsf{KGen}(1^\lambda, \mathsf{max})$
$\alpha, \chi \leftarrow\$ \mathbb{F}_p^2$
return $\left[\{\chi^i\}_{i=-\mathsf{n}}^{\mathsf{n}}, \{\alpha\chi^i\}_{i=-\mathsf{n}, i\neq 0}^{\mathsf{n}}\right]_1,$
    $\left[\{\chi^i, \alpha\chi^i\}_{i=-\mathsf{n}}^{\mathsf{n}}\right]_2, [\alpha]_T$

$\mathsf{Com}(\mathsf{srs}, \mathsf{max}, \mathsf{f}(X))$
$\mathsf{c}(X) \leftarrow \alpha \cdot X^{\mathsf{d}-\mathsf{max}} \mathsf{f}(X)$
return $[c]_1 = [\mathsf{c}(\chi)]_1$

$\mathsf{Op}(\mathsf{srs}, z, s, f(X))$
$\mathsf{o}(X) \leftarrow \frac{\mathsf{f}(X) - \mathsf{f}(z)}{X - z}$
return $[\mathsf{o}(\chi)]_1$

$\mathsf{Vf}(\mathsf{srs}, \mathsf{max}, [c]_1, z, s, [\mathsf{o}(\chi)]_1)$
if $[\mathsf{o}(\chi)]_1 \bullet [\alpha\chi]_2 + [s - z\mathsf{o}(\chi)]_1 \bullet [\alpha]_2 =$
    $[c]_1 \bullet \left[\chi^{-\mathsf{d}+\mathsf{max}}\right]_2$ then return $1$
else return $0$.
```

Fig. 4: $\mathsf{PC_S}$ polynomial commitment scheme.

References

1. 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 20*, pages 1987–2005. ACM Press, Nov. 2020.
2. 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`.
3. 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`.
4. A. Bagherzandi, J. H. Cheon, and S. Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In P. Ning, P. F. Syverson, and S. Jha, editors, *ACM CCS 2008*, pages 449–458. ACM Press, Oct. 2008.
5. M. Bellare, W. Dai, and L. Li. The local forking lemma and its application to deterministic encryption. In S. D. Galbraith and S. Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 607–636. Springer, Heidelberg, Dec. 2019.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. 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.
15. 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`.
16. 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.

17. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. https://eprint.iacr.org/2000/067.
18. 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.
19. 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.
20. 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.
21. 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.
22. 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.
23. L. Fortnow. The complexity of perfect zero-knowledge (extended abstract). In A. Aho, editor, *19th ACM STOC*, pages 204–209. ACM Press, May 1987.
24. 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.
25. 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.
26. 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.
27. A. Ghoshal and S. Tessaro. Tight state-restoration soundness in the algebraic group model. Cryptology ePrint Archive, Report 2020/1351, 2020. https://eprint.iacr.org/2020/1351.
28. 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.
29. 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.
30. 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.
31. 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.
32. 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.
33. 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.

34. 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.
35. J. Herranz and G. Sáez. Forking lemmas for ring signature schemes. In T. Johansson and S. Maitra, editors, *INDOCRYPT 2003*, volume 2904 of *LNCS*, pages 266–279. Springer, Heidelberg, Dec. 2003.
36. J. Holmgren. On round-by-round soundness and state restoration attacks. Cryptology ePrint Archive, Report 2019/1261, 2019. `https://eprint.iacr.org/2019/1261`.
37. 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.
38. 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`.
39. 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.
40. 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.
41. H. Lipmaa. Key-and-argument-updatable QA-NIZKs. Cryptology ePrint Archive, Report 2019/333, 2019. `https://eprint.iacr.org/2019/333`.
42. 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.
43. S. Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, Nov. 1994.
44. 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.
45. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000.
46. C.-P. Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, Heidelberg, Aug. 1990.
47. V. Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. `https://eprint.iacr.org/2004/332`.