Non-Malleability of the Fiat–Shamir Transform Revisited for Multi-round SRS-Based Protocols

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

[1] University of Edinburgh, Edinburgh, UK
[2] IOHK
mkohlwei@inf.ed.ac.uk
[3] Clearmatics, London, UK
m.p.zajac@gmail.com

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 prove 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 protocol 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 past and present SRS. 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 update 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 [**?**, **?**, **?**]. 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 [**?**], 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 [**?**] in Zcash and scalable and private smart contracts in Ethereum[4].

The work of [**?**] 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 [**?**,**?**,**?**] and led to several works [**?**,**?**,**?**,**?**,**?**, **?**] 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. 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 [**?**], Plonk [**?**], Marlin [**?**] 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

[4] https://z.cash/,https://ethereum.org

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 [**?**, **?**, **?**, **?**, **?**, **?**, **?**] 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 [**?**] 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, [**?**] 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 [**?**]. Subsequently, Gabizon et al. designed Plonk [**?**] which currently is the most efficient updatable universal zkSNARK. Independently, Chiesa et al. [**?**] 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 [**?**, **?**] 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) [**?**], it is theoretically unsound [**?**] 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 [**?**] Plonk [**?**], and Marlin [**?**] 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.
- *General theorem for USE of FS-compiled interactive protocols.* We then show that a class of interactive proofs of knowledge that are honest-verifier zero-knowledge, have a unique response property, and satify 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 [**?**]. However, we need new and stronger notions as we consider proof systems that are richer than simple sigma protocols. We discuss some of the technical challenges below.

Plonk [**?**] and Sonic [**?**] 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) and S (for Sonic) and the resulting non-interactive proof systems by P_{FS} and S_{FS}, respectively.

Forking soundness. Following [**?**], 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 SRS. For P and S 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 P and $(\mathsf{n}+\mathsf{Q}+1)$—where n and Q are the numbers of multiplicative and linear constraints—for S. Hence, we do not have a pair of transcripts, but a *tree of transcripts*.

In protocols that rely on SRS that come with a trapdoors, an adversary in posession 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.

Unique response property. Another property required to show USE is the unique response property [**?**] 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 P_{FS}, both the first and the second prover's messages are randomized. Although S prover is deterministic after it picks its first message, the protocol has more than 3 rounds. 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 P for $i = 2$. Since Sonic's prover is deterministic from the second message on, it is 1-ur.

HVZK. In order to invoke our main theorem on P_{FS} and S_{FS} to conclude USE, we also need to show that (interactive) P and S are HVZK in the standard model. Although both Sonic and Plonk are zero-knowledge, their simulators utilize trapdoors. However, 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 wat. (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} and S_{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?

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 [**?**], 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 [**?**, **?**] 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 [**?**] 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 [**?**] noticed that a (black-box) SE NIZK is universally-composable (UC) [**?**]. Then Dodis et al. [**?**] 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 [**?**] and SE zkSNARK for QAP by Lipmaa [**?**]. Kosba's et al. [**?**] 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. [**?**] showed another transformation that obtains non-black-box simulation extractability but also preserves succinctness of

the argument. The zkSNARK of [**?**] has been shown to be SE by introducing minor modifications to the construction and making stronger assumptions [**?**, **?**]. Recently, [**?**] showed that the original Groth's proof system from [**?**] 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., [**?**, **?**, **?**], of the general forking lemma [**?**, **?**] for analyzing the forking behavior of random-oracle based executions. In [**?**], 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. [**?**, **?**]. 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 [**?**, **?**] 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 [**?**] 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 partiular, 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

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 $\Delta(X_\lambda, Y_\lambda) \leq \mathsf{negl}(\lambda)$. For values $a(\lambda)$ and $b(\lambda)$ we write $a(\lambda) \approx_\lambda b(\lambda)$ if $|a(\lambda) - b(\lambda)| \leq \mathsf{negl}(\lambda)$.

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. $\overline{\mathsf{E}} = \Omega \setminus \mathsf{E}$.

Denote by $\mathcal{R} = \{\mathbf{R}\}$ a family of relations. We assume that if $\mathbf{R}$ comes with any auxiliary input, it is benign. Directly from the description of $\mathbf{R}$ one learns security parameter λ and other necessary information like public parameters p containing description of a 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.

2.1 Computational assumptions.

Discrete-log assumptions. Security of Plonk and Sonic relies on two discrete-log based security assumptions—(q_1, q_2)-dlog assumption and its variant that allows for negative exponents (q_1, q_2)-ldlog assumption[5]. We omit here description of the assumptions and refer to Appendix A.1.

Proofs by Game-Hopping. Proofs by *game hopping* is a method of writing proofs popularised by e.g. Shoup [**?**] and Dent [**?**]. The method relies on the following lemma.

Lemma 1 (Difference lemma, [?, 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* $|\Pr[\mathsf{A}] - \Pr[\mathsf{B}]| \leq \Pr[\mathsf{F}]$.

2.2 Algebraic Group Model

The algebraic group model (AGM) introduced in [**?**] lies between the standard model and generic bilinear group model[**Hamid:** not clear why generic bilinear group model and not just generic 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*.

2.3 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 [**?**]. We denote the first by PC_P and the latter by PC_S. Due to page limit, we omit their presentation here and refer to Fig. 5 and Fig. 6 in the Appendix B.1. 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 Appendix B.1, 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.

[5] Note that [**?**] dubs their assumption *a dlog assumption*. We changed that name to distinguish it from the more standard dlog assumption used in [**?**]. "l" in *ldlog* relates to use of Laurent polynomials in the assumption.

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. [**?**] shows, using AGM, that PC_{S} is a commitment of knowledge. The same reasoning could be used to show that property for 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 5 and Lemma 9 respectively.

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

2.4 Zero knowledge

In a zero-knowledge proof system, a prover convinces the verifier of veracity of a statement without leaking any other information. The zero-knowledge property is proven by constructing a simulator that can simulate the view of a cheating verifier without knowing the secret information—witness—of the prover. A proof system has to be sound as well, i.e. for a malicious prover it should be infeasible to convince a verifier on a false statement. Here, we focus on proof systems that guarantee soundness against PPT malicious provers.

More precisely, 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 and V be PPT algorithms, the former called *prover* and the latter *verifier*. 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. We sometimes abuse notation and write $\mathsf{V}(\mathsf{srs}, \mathsf{x}, \pi) = 1$ to denote a fact that π is accepted by the verifier. (This is especially handy when the proof system is non-interactive, i.e. the whole conversation between the prover and verifier consists of a single message π sent by P).

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 $\mathbf{R} \in \mathcal{R}(1^\lambda)$

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

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 $(\mathsf{x}, \mathsf{w}) \in \mathbf{R}$

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

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 ana 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* $b = 0q$ *or a simulated proof* $\pi_{\mathsf{Sim}} \leftarrow \mathsf{Sim}(\mathsf{srs}, \mathsf{x})$ *if* $b = 1$*.*

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 Lemmas 8 and 12 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 $[\mathsf{p}_1(\chi), \ldots, \mathsf{p}_k(\chi)]_1$, for some polynomials $\mathsf{p}_1(X), \ldots, \mathsf{p}_k(X) \in \mathbb{F}_p[X]$. 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 [**?**] 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

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

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

Another property that sigma protocols may have is a unique response property [**?**] 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,:

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

If this property holds even against unbounded adversaries, it is called *strict*, cf. [**?**]. 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 [**?**], fulfil this property.

2.5 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, $i \in [1\,..\,\mu]$, 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 $\mathbf{R} \in \mathcal{R}(1^\lambda)$.

2.6 Non-malleability definitions for NIZKs

Real life applications often require a NIZK proof system to be non-malleable. That is, no adversary seeing a proof π for a statement x should be able to provide a new proof π' related to π. *Simulation extractability* formalizes a strong version of non-malleability by requiring that no adversary can produce a valid proof without knowing the corresponding witness. This must hold even if the adversary is allowed to see polynomially many simulated proofs for any statements it wishes.

Definition 3 (Forking simulation-extractable NIZK, [?]). *Let $\Psi_{\mathsf{FS}} = (\mathsf{KGen}_{\mathsf{FS}}, \mathsf{P}_{\mathsf{FS}}, \mathsf{V}_{\mathsf{FS}}, \mathsf{Sim}_{\mathsf{FS}})$ be a HVZK proof system[***Hamid:*** Ψ_{FS} *is the Fiat-Shamir variant of the underlying proof system. So maybe we mean the underlying proof system is HVZK?]. We say that Ψ_{FS} is* forking simulation-extractable *with* extraction error ν *if for any* PPT *adversary $\mathcal{A}$ that is given oracle access to a random oracle $\mathcal{H}$ and simulator $\mathsf{Sim}_{\mathsf{FS}}$, and produces an accepting transcript of Ψ with probability* acc*, where*

$$\mathsf{acc} = \Pr\left[\begin{array}{l|l} \mathsf{V}_{\mathsf{FS}}(\mathsf{srs}, \mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}) = 1, & \mathsf{srs} \leftarrow \mathsf{KGen}_{\mathsf{FS}}(\mathbf{R}), r \leftarrow_{\$} \mathsf{R}(\mathcal{A}), \\ (\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}) \notin Q & (\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathsf{Sim}_{\mathsf{FS}}, \mathcal{H}}(\mathsf{srs}; r) \end{array}\right],$$

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

$$\mathsf{ext} = \Pr\left[\begin{array}{l|l} \mathsf{V}_{\mathsf{FS}}(\mathsf{srs}, \mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}) = 1, & \mathsf{srs} \leftarrow \mathsf{KGen}_{\mathsf{FS}}(\mathbf{R}), r \leftarrow_{\$} \mathsf{R}(\mathcal{A}), \\ (\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}) \notin Q, & (\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathsf{Sim}_{\mathsf{FS}}, \mathcal{H}}(\mathsf{srs}; r) \\ \mathbf{R}(\mathsf{x}_{\mathcal{A}}, \mathsf{w}_{\mathcal{A}}) = 1 & \mathsf{w}_{\mathcal{A}} \leftarrow \mathsf{Ext}_{\mathsf{se}}(\mathsf{srs}, \mathcal{A}, r, \mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}, Q, Q_{\mathcal{H}}) \end{array}\right]$$

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* $\mathsf{acc} \geq \nu$*. 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.*

$\mathsf{GF}^m_{\mathcal{Z}}(y, h^1_1, \ldots, h^1_q)$

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

Fig. 1: Generalised forking algorithm $\mathsf{GF}^m_{\mathcal{Z}}$

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

[**Chaya:** move the USE definition to this section?]

The result of Faust et al. [**?**] do not apply to our setting since the protocols we consider have an SRS, more than three messages, require more than just two transcripts for standard model extraction and are not special sound. We thus adapt special soundness to forking soundness, and generalize the forking lemma and the unique response property to make them compatible with multi-round SRS-based protocols.

3.1 Generalised forking lemma.

First of all, although dubbed "general", Lemma 16 is not general enough for our purpose as it is useful only for protocols where witness can be extracted from just two transcripts. To be able to extract a witness from, say, an execution of P we need to obtain at least $(3\mathsf{n} + 1)$ valid proofs, and $(\mathsf{n} + \mathsf{Q} + 1)$ for S. Here 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. [?]). *Consider a $(2\mu + 1)$-message interactive proof system Ψ. A $(n_1, \ldots, n_\mu)$-tree of accepting transcripts is a tree where each node on depth i, for $i \in [1\,..\,\mu + 1]$, is an i-th prover's message in an accepting transcript; edges between the nodes are labeled with 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 $y, h_1, \ldots, h_q$ returns (i, s) where $i \in [0\,..\,q]$ and s is called a side output. Denote by IG a randomised instance generator. We denote by acc the probability*

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

Let $\mathsf{GF}^m_{\mathcal{Z}}$ denote the algorithm described in Fig. 1 then the probability $\mathsf{frk} := \Pr\left[b = 1 \;\middle|\; y \leftarrow \mathsf{IG};\ h_1, \ldots, h_q \leftarrow\!\!\$\ H;\ (b, s) \leftarrow \mathsf{GF}^m_{\mathcal{Z}}(y, h_1, \ldots, h_q)\right]$ 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 goes similarly to [**?**, Lemma 1] with some modifications required by the fact that the protocol has more than 3 rounds and the number of transcripts required is larger. Due to page limit, the proof is presented in Appendix A.4.

To highlight importance of the generalised forking lemma we describe how we use it 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 an $(1, \ldots, 1, n_k, 1, \ldots, 1)$-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 $\mathcal{Z}$ outputing a non-zero i and the probability of $\mathcal{A}$ outputing an accepting proof, we claim that these probabilities are exactly the same by the way we define $\mathcal{Z}$.) Let $\mathcal{A}$ produce an accepting proof $\pi_{\mathcal{A}}$ for instance $\mathsf{x}_{\mathcal{A}}$; r be $\mathcal{A}$'s randomness; Q the list of queries submitted by $\mathcal{A}$ along with simulator 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. Fig. 4, gets only as input 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 proposed in 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.

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 [**?**, **?**], 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:

Definition 5 (i-ur-protocol). *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* $\mathcal{A}$*:*

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

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

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. [**Chaya:** a notion of computational special soundness has been used to mean exactly the above in prior works, like BBF19. we should clarify if forking soundness different from computational special soundness? seems to me like the diference is just that here it is tailored for the NI version.] [**Chaya:** I now see that the diff is the access to the simulator that the adversary gets. might be helpful to clarify this, either here or in a tech overview section. I am not sure why this needs to be defined this way by giving access to the simulator.] 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).** *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* $\mathcal{A}^{\mathsf{Sim}_{\mathsf{FS}}, \mathcal{H}}(\mathsf{srs}; r)$ *we consider the procedure* $\mathcal{Z}$ *that provided the transcript* $(\mathsf{srs}, \mathcal{A}, r, Q, Q_H)$ *and* $h_1, \ldots, h_q$ *runs* $\mathcal{A}$ *by providing it with random oracle queries and simulated proofs. While* Q_H *is consistent with* $h_1, \ldots, h_q$*, it replays the proofs of* Q*.* $\mathcal{Z}$ *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 such that none of the* n *challenges in round* k *were used in simulated proofs.*

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

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

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.*

Definition 7 **($(\varepsilon(\lambda), k, n)$-forking soundness).** *Let* $\Psi = (\mathsf{KGen}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *be an* $(2\mu+1)$*-message proof system for a relation* $\mathbf{R}$.

Let $\mathcal{T}$*, called tree creator, be the algorithm below that rewinds the* PPT *adversary* $\mathcal{A}^{\mathsf{Sim}_{\mathsf{FS}}, \mathcal{H}}(\mathsf{srs}; r)$ *to produce a* $(1, \ldots, n, \ldots, 1)$*-tree of transcripts such that none of the* n *challenges in round* k *were used in simulated proofs.*

$\mathcal{T}$ *has oracle access to* $\mathcal{A}$ *and provides it with (oracle) access to random oracle* $\mathcal{H}$ *and simulator* $\mathsf{Sim}_{\mathsf{FS}}$ *– more precisely* $\mathcal{T}$ *has an internal procedure* $\mathcal{B}$ *that provided* srs *and random oracle queries' responses* $h_1, \ldots, h_q$ *gives* $\mathcal{A}$ *access to the random oracle and simulates proof for it. In the end,* $\mathcal{B}$

returns the index i of the random oracle query made for challenge k, the set Q of simulator random oracle indexes, the instance x, and the proof $\mathcal{A}$ returns. Eventually, $\mathcal{T}$ returns a $(1, \ldots, n, \ldots, 1)$ tree of acceptable transcripts T.

$\mathcal{T}(\mathcal{A}, \mathsf{srs} \leftarrow_\$ \mathsf{KGen}(\mathbf{R}))$

$h_1^1, \ldots, h_q^1 \leftarrow_\$ H$
$(i, Q, \mathsf{x}, \pi_1) \leftarrow \mathcal{B}(\mathcal{A}, \mathsf{srs}, h_1^1, \ldots, h_q^1)$
if $i \in Q \vee \mathsf{V}(\mathsf{srs}, \mathsf{x}, \pi_1) = 0$ **return** $(0, \perp)$
for $j \in [2 \,..\, m]$
  $h_1^j, \ldots, h_{i-1}^j \leftarrow h_1^{j-1}, \ldots, h_{i-1}^{j-1}$
  $h_i^j, \ldots, h_q^j \leftarrow_\$ H$
  $(i_j, Q_j, \mathsf{x}_j, \pi_j) \leftarrow \mathcal{B}(\mathcal{A}, \mathsf{srs}, h_1^j, \ldots, h_{i-1}^j, h_i^j, \ldots, h_q^j)$
  if $i \neq i_j \vee i_j \in Q_j \vee \mathsf{x} \neq \mathsf{x}_j \vee \mathsf{V}(\mathsf{srs}, \mathsf{x}_j, \pi_i) = 0$ **return** $(0, \perp)$
else return $(1, \mathsf{T} = (\mathsf{x}, \boldsymbol{\pi}))$

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

$$\Pr\left[\mathsf{w} \leftarrow \mathsf{Ext}_{\mathsf{tree}}(\mathsf{T}), \mathbf{R}(\mathsf{x}, \mathsf{w}) = 0 \mid \mathsf{srs} \leftarrow_\$ \mathsf{KGen}(\mathbf{R}), (1, \mathsf{T}) \leftarrow \mathcal{T}(\mathcal{A}, \mathsf{srs})\right] \leq \varepsilon(\lambda).$$

4 Updatable Simulation Extractable SNARKs

Updatable SRS setting. Let us recall the definition of an updatable SRS scheme from [**?**] 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.

Definition 8. *An updatable SRS scheme is correct if the following completeness properties are satisfied.*

- *For all $\lambda \in \mathbb{N}$*
$$\Pr\left[\mathsf{VerifySRS}(1^\lambda, \mathsf{srs}, \rho) = 1 \;\middle|\; (\mathsf{srs}, \rho) \leftarrow \mathsf{KGen}(1^\lambda)\right] = 1$$
- *For all $(\mathsf{srs}, \{\rho_i\}_{i=1}^n)$ such that $\mathsf{VerifySRS}(1^\lambda, \mathsf{srs}, \{\rho_i\}_{i=1}^n)) = 1$,*
$$\Pr\left[\mathsf{VerifySRS}(1^\lambda, \mathsf{srs}', \{\rho_i\}_{i=1}^{n+1})) = 1 \;\middle|\; (\mathsf{srs}', \rho_{n+1}) \leftarrow \mathsf{Upd}(1^\lambda, \mathsf{srs}, \{\rho_i\}_{i=1}^n)\right] = 1$$

Intuitively, the updatability of the SRS should allow an adversarial prover to contribute to updating, and see proofs with respect to different updated SRS'es before attempting to provide a proof for a false statement. Note that, the adversary's output could be a proof w.r.t. a SRS that is different from the SRSs corresponding to all the simulated proofs seen. In addition, since the SRS is universal, the adversary could ask for a simulated proof for different statements as well. We think of the instance x as consisting of the relation as well as the public input. This is without loss of generality since the SRS specialization for a relation is deterministic, and derived from an updated universal SRS. In concrete zkSNARKs, there is a preprocessing phase that corresponds to this specialization, which we hide in the definition by absorbing the relation into the instance.

We first formalize an intermediate notion of updatable simulation extractability (USE) called *late updatable simulation extractability* (LUSE), where the adversary sees simulated proofs only with respect

to the last SRS. We then give our definition of USE that models a real-world adversary capable of seeing simulated proofs with respect to different (finalized) SRS'es.

We then prove a lemma that relates USE to LUSE. This lemma provides a clean and modular approach for proving USE property of zkSNARKs. For the sake of completeness, we also recall the definitions of standard simulation extractability and updatable knowledge soundness. In Fig. 2, 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, and Q_{srs} contains proofs of all honest SRS updates.

Definition 9 (Updatable knowledge soundness [?]). *An argument system for a relation* $\mathbf{R}$ *is updatable knowledge-sound if for all PPT algorithms* $\mathcal{A}$*, there exists an extractor* $\mathsf{Ext}_{\mathsf{uks}}$ *such that the advantage* $\mathsf{Adv}_{\mathsf{uks}} := \Pr\left[\mathsf{UKS}_{\mathcal{A},\mathsf{Ext}_{\mathsf{uks}}}(\lambda) = 1\right]$ *of* $\mathcal{A}$ *in the game defined in Fig. 2 is negligible in* λ*.*

Definition 10 (Simulation Extractability). *An argument system for a relation* $\mathbf{R}$ *is simulation-extractable if for all PPT algorithms* $\mathcal{A}$*, there exists an extractor* $\mathsf{Ext}_{\mathsf{se}}$ *such that the advantage* $\mathsf{Adv}_{\mathsf{se}} := \Pr\left[\mathsf{SE}_{\mathcal{A},\mathsf{Ext}_{\mathsf{se}}}(\lambda) = 1\right]$ *of* $\mathcal{A}$ *in the game defined in Fig. 2 is negligible in* λ*.*

Definition 11 (Late Updatable Simulation Extractability). *An argument system for a relation* $\mathbf{R}$ *is late updatable simulation-extractable if for all PPT algorithms* $\mathcal{A}$*, there exists an extractor* $\mathsf{Ext}_{\mathsf{luse}}$ *such that the advantage* $\mathsf{Adv}_{\mathsf{luse}} := \Pr\left[\mathsf{LUSE}_{\mathcal{A},\mathsf{Ext}_{\mathsf{luse}}}(\lambda) = 1\right]$ *of* $\mathcal{A}$ *in the game defined in Fig. 2 is negligible in* λ*.*

Definition 12 (Updatable Simulation Extractability). *An argument system for a relation* $\mathbf{R}$ *is updatable simulation-extractable if for all PPT algorithms* $\mathcal{A}$*, there exists an extractor* $\mathsf{Ext}_{\mathsf{use}}$ *such that the advantage* $\mathsf{Adv}_{\mathsf{use}} := \Pr\left[\mathsf{USE}_{\mathcal{A},\mathsf{Ext}_{\mathsf{use}}}(\lambda) = 1\right]$ *of* $\mathcal{A}$ *in the game defined in Fig. 2 is negligible in* λ*.*

The following lemma relates the definition of updatable simulation extractability to the definition of late updatable simulation extractability and updatable knowledge soundness. More specifically, the lemma shows that for any PPT adversary $\mathcal{A}$ that has a non-negligible advantage in $\mathsf{USE}_{\mathcal{A},\mathsf{Ext}_{\mathsf{use}}}(\lambda)$, there exists a PPT adversary $\mathcal{B}$ that can win with non-negligible advantage either $\mathsf{LUSE}_{\mathcal{A},\mathsf{Ext}_{\mathsf{luse}}}(\lambda)$ or $\mathsf{UKS}_{\mathcal{A},\mathsf{Ext}_{\mathsf{uks}}}(\lambda)$.

Lemma 3. *Let* π *be a proof system that satisfies updatable knowledge soundness and late updatable simulation extractability. Then,* π *satisfies updatable simulation extractability.*

Proof. Let $\mathcal{A}$ be an adversary against updatable simulation extractability. We construct an adversary $\mathcal{B}$ that breaks either updatable knowledge soundness or late updatable simulation extractability. The key idea is to construct adversaries $\mathcal{B}_0$ and $\{\mathcal{B}_1, \mathcal{B}_2 \ldots\}$ against updatable knowledge soundness and against late updatable simulation extractability respectively. In more detail, every time $\mathcal{A}$ makes a simulation query with respect to a SRS srs_i, the adversary $\mathcal{B}$ plays the role of a late updatable simulation extractable adversary and forwards the same statement to its own simulator, after finalizing srs_i. That is, whenever $\mathcal{B}$ receives a query $(\mathsf{srs}_i, \mathcal{P}_i, \mathsf{x})$ from $\mathcal{A}$ who asks for a simulated proof with respect to SRS srs_i (with $\mathcal{P}_i$ being the set of its update proofs) and statement x, it does the following:

- $\mathcal{B}$ first checks if the conditions $\mathsf{VerifySRS}(1^\lambda, \mathsf{srs}_i, \mathcal{P}_i) = 1$ and $Q \cap \mathcal{P}_i \neq \emptyset$ hold. The first condition checks if srs_i is well formed. The second condition verifies if $\mathcal{P}_i$ contains at least one honest update proof, where Q is the set of all honest update proofs maintained by $\mathcal{B}$.
- If the conditions verify, it plays the role of $\mathcal{B}_i$, and finalizes srs_i by calling the update oracle with intent "`final`".
- $\mathcal{B}_i$ then makes a query x to its own simulation oracle with respect to the finalized SRS in the corresponding simulation extractability game.
  Note that for $\mathcal{B}$ to be able to engage in the i-th updatable SE game with respect to the fresh SRS srs_i and get access to a simulator for srs_i, it needs to perform all previous updates up to that point and finalize srs_i. To this end, $\mathcal{B}$ internally keeps track of all the updates $\mathcal{A}$ has made up to i, including any honest updates.

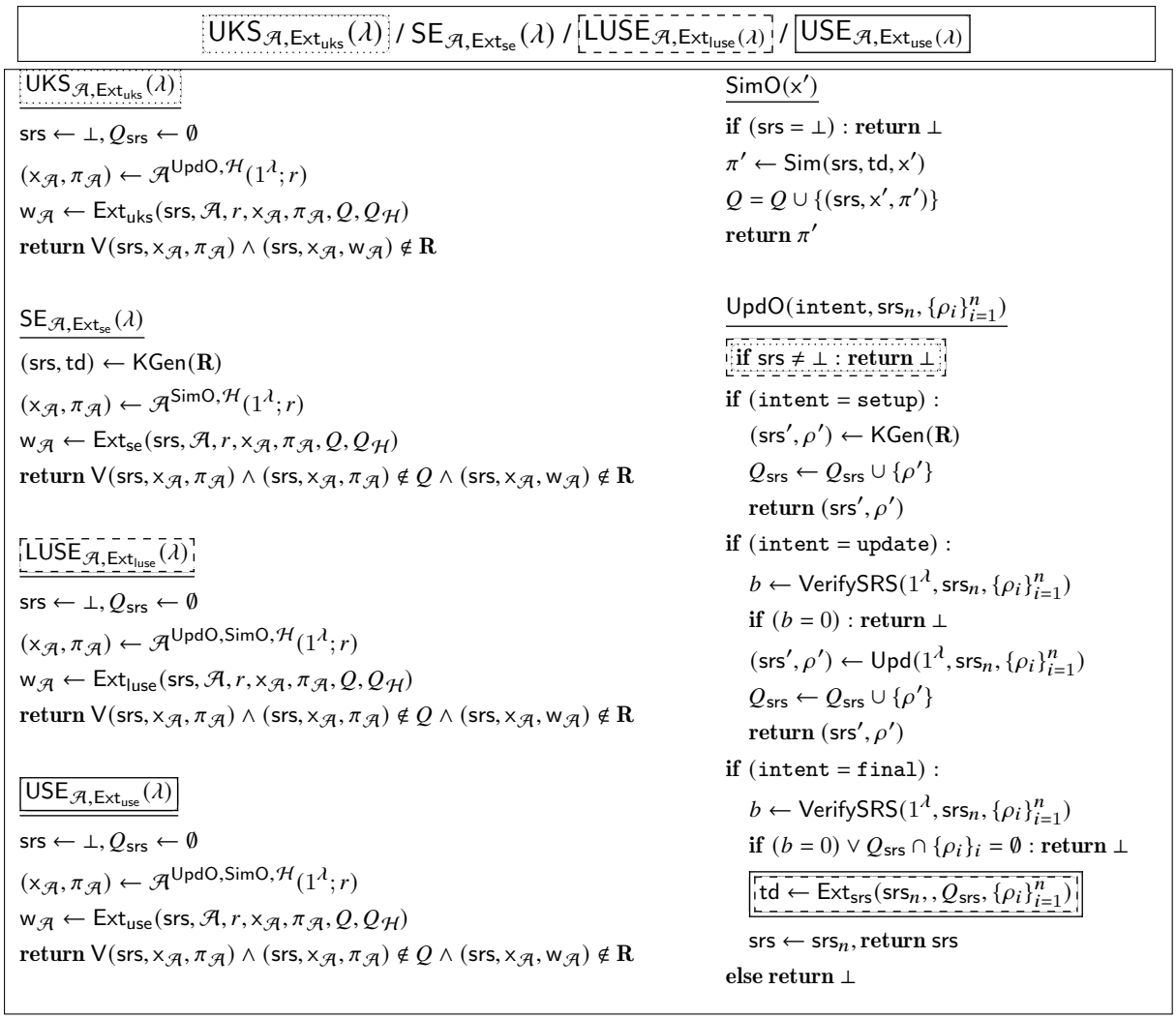

Fig. 2: Games in the Updatable SRS setting. The boxed/dashed-box/dotted-box part is only present in the boxed/dashed-boxed/dotted-boxed game.

– Upon receiving the proof π from its simulation oracle, $\mathcal{B}_i$ returns π to $\mathcal{A}$.

[**Chaya:** we are assuming that once the adversary updates the srs, it loses access to the simulation oracle wrt the previous srs. It can only see proofs already queried before the updation. Discuss if this is reasonable. may not even be necessary.][**Hamid:** The current argument is with respect to this (restricted) definition, but such restriction seems natural, because even a real-world adversary cannot control provers to generate proofs with respect to outdated SRSs.]

Let us assume that srs_ℓ is the last distinct SRS updated by $\mathcal{A}$. At the end of the game, $\mathcal{A}$ returns $(\mathsf{srs}^*, \mathsf{x}^*, \pi^*)$, and since $\mathcal{A}$ wins, $\mathsf{srs}^* = \mathsf{srs}_\ell$. Now, $\mathcal{B}$ checks whether $\mathcal{A}$ previously made a simulation query with srs^*, that is, whether $\mathcal{B}_\ell$ was invoked. If so, it plays the role of $\mathcal{B}_\ell$ and returns (x^*, π^*) to the challenger in the corresponding simulation extractability game. Otherwise, if srs^* is an SRS for which $\mathcal{A}$ made no simulation queries, then $\mathcal{B}$ plays the role of $\mathcal{B}_0$ and returns $(\mathsf{srs}^*, \mathsf{x}^*, \pi^*)$ to its updatable knowledge soundness challenger. It is easy to see that if $\mathcal{A}$ has a non-negligible advantage, then $\mathcal{B}$, as constructed above, also has the same advantage (with $1/k$ loss where k is total number of updates made by $\mathcal{A}$) in breaking either updatable knowledge soundness or late updatable simulation extractability. □

5 From a late updatable SE to SE

Michal 13.09: Here I want to show an idea of a reduction of a late updatable SE to SE

Let Plonk be a forking simulation extractable proof system. Let $\mathcal{B}$ be a SE adversary. Let $\mathcal{A}$ be an *algebraic* LUSE adversary. We show that existence of an extractor for $\mathcal{B}$, $\mathsf{Ext}_{\mathcal{B}}$, assures existence of $\mathsf{Ext}_{\mathcal{A}}$ – extractor for $\mathcal{A}$.

Here we simplify Plonk a bit and assume that polynomials are written in standard bases instead of Lagrange. That is, for instance, $\mathsf{a}(X) = \sum_{i=0}^{n} a_i X^i + \mathsf{Z_H}(X)(b_1 + b_2 X)$

The argument goes as follows. Let $\mathsf{srs}_0 \rightarrow \mathsf{srs}_1 \rightarrow \ldots \mathsf{srs}_n$ be the sequence of SRS co-produced by $\mathcal{A}$. Let srs_k be the last SRS honestly computed.

The adversary $\mathcal{B}$ proceeds as follows:

1. Get as input an SRS srs honestly created using some trapdoor χ.
2. Internally run adversary $\mathcal{A}$. Process $\mathcal{A}$'s SRS update queries.
3. Guess index k of the last honest update, set $\mathsf{srs}_k = \mathsf{srs}$. (From now on we will denote this SRS by srs.)
4. Let $\mathcal{A}$ update srs according to its wishes. Let's the final SRS be srs_n computed using some trapdoor $\alpha\chi$. (α picked by $\mathcal{A}$)
5. Since $\mathcal{A}$ is algebraic, $\mathcal{B}$ learns α.
6. $\mathcal{B}$ processes $\mathcal{A}$ simulation queries (that should be done according to srs_n).
   (a) On instance and witness w' (note that in Plonk the instance is a part of the witness) compute w such that $\mathsf{w}_i = \alpha^i \mathsf{w}_i$, for $i = 1..(n-1)$; ($\mathcal{B}$ gives as instance/witness the coefficients of some polynomial $\tilde{\mathsf{a}}'$ which commitment is its evaluation at $\alpha\chi$. We need to express it the basis related to χ as the obtained polynomial $\tilde{\mathsf{a}}$ will be committed at χ. We observe $\tilde{\mathsf{a}}'(\alpha\chi) = \tilde{\mathsf{a}}(\chi)$, for some $\tilde{\mathsf{a}}$ we are now computing) $\mathsf{w}_{i+n} = \alpha^i \mathsf{w}_{i+n}$, $\mathsf{w}_{i+2n} = \alpha^i \mathsf{w}_{i+2n}$ (the last two translations are since we need to compute $\tilde{\mathsf{b}}$ from $\tilde{\mathsf{b}}'$ and $\tilde{\mathsf{c}}$ from $\tilde{\mathsf{c}}'$)
   (b) Provide w to the simulator.
   (c) Get simulated proof $[\mathsf{a}(\chi), \mathsf{b}(\chi), \mathsf{c}(\chi)]_1$, β, γ, $\mathsf{z}(\chi)$, α, $\mathsf{t}(\chi)$, $\mathfrak{z}$, $\mathsf{a}(\mathfrak{z}), \ldots, \mathsf{t}(\mathfrak{z}), \mathsf{z}(\mathfrak{z}\omega)$, δ. The simulator (which uses trapdoor, that's fine here and simplifies things) simply takes random polynomials $\mathsf{a}, \mathsf{b}, \mathsf{c}$, random challenges, compute z, t in regard with the picked polynomials and challenges. Commits to polynomials. Sim uses trapdoor to commit to t (it is infeasible for an adversary which doesn't know the instance's witness.)
   (d) Translate the proof.
      i. Since simulator for $\mathcal{A}$ would pick random polynomials $\mathsf{a}', \mathsf{b}', \mathsf{c}'$ and send $[\mathsf{a}'(\alpha\chi), \mathsf{b}'(\alpha\chi), \mathsf{c}'(\alpha\chi)]_1$. Just send $\mathsf{a}(\chi), \mathsf{b}(\chi), \mathsf{c}(\chi)$. That is $\mathsf{a}'(\alpha\chi) = \mathsf{a}(\chi)$ and $\mathsf{a}'(\alpha X) = a'_0 + a'_1(\alpha X) + \ldots + a'_n(\alpha^n X^n) = a_0 + a_1\alpha X + \ldots + (a_n\alpha^n)X^n$. Output commitments to primed polynomials.
      ii. The polynomials are the same, hence RO answers β, γ are the same as well.
      iii. Polynomial z can also be picked by the simulator at random, hence $\mathsf{z}'(\alpha\chi) := \mathsf{z}(\chi)$. Output $[\mathsf{z}'(\alpha\chi)]_1$
      iv. Again, the random oracle response is the same: α.
      v. Since $\mathsf{t}'(X)$ is determined by $\mathsf{a}'(X), \mathsf{b}'(X), \mathsf{c}'(X), \mathsf{z}'(X)$ and some publicly known polynomials, and $[\mathsf{a}'(\alpha\chi)]_1 = [\mathsf{a}(\chi)]_1, \ldots$ we set $\mathsf{t}'(\alpha\chi) = \mathsf{t}(\chi)$ and output $[\mathsf{t}'(\alpha\chi)]_1$.
      vi. Get evaluation challenge $\mathfrak{z}$ and compute $\mathfrak{z}' = \mathfrak{z}\alpha$ (need to adjust evaluation point to have the same evaluation values for primed and non-primed polynomials).
      vii. Get evaluations $\mathsf{a}(\mathfrak{z}), \ldots$; output them as evaluations $\mathsf{a}'(\mathfrak{z}'), \ldots$.
      viii. Get the opening challenge ν *program $\mathcal{A}$'s oracle to output* ν Michal 13.09: we changed the partial transcript hence we need to program the oracle. Is that a problem?
      ix. Compute evaluations' openings. Observe that (here simplification – we show correctness of evaluation to a but batched version should work similarly)

$$W_{\mathfrak{z}}(\chi) = \frac{\mathsf{a}(\chi) - \mathsf{a}(\mathfrak{z})}{\chi - \mathfrak{z}} = \tag{2}$$

$$\frac{\mathsf{a}'(\alpha\chi) - \mathsf{a}'(\alpha\mathfrak{z})}{\chi - \mathfrak{z}} = \tag{3}$$

$$\alpha\frac{\mathsf{a}'(\alpha\chi) - \mathsf{a}'(\alpha\mathfrak{z})}{\alpha\chi - \alpha\mathfrak{z}} \tag{4}$$

$$= \alpha W_{\mathfrak{z}\alpha}(\alpha\chi). \tag{5}$$

Hence $\mathcal{B}$ sends to $\mathcal{A}$ opening $W_{\mathfrak{z}\alpha}(\alpha\chi) = \frac{1}{\alpha}W_{\mathfrak{z}}(\chi)$.

(e) Also, the verification of the correctness of the opening holds. Michal 13.09: Note, for the batched version we need to query random oracle to get batching coefficients. Here we need to program RO again, to have the same coeffs.

7. Now we need to show that proof output by $\mathcal{A}$ for srs_n can be translated to a proof in srs. This is done similarly to the above. All RO queries $\mathcal{A}$ makes to create the proof are answered honestly by $\mathcal{B}$ except the query that gives challenge $\mathfrak{z}'$. More precisely, when $\mathcal{A}$ passes a partial transcript produced by $\mathcal{A}$ to get $\mathfrak{z}'$ it gets RO's answer $\mathfrak{z}$ and set $\mathfrak{z}' = \alpha\mathfrak{z}$.
Unique response property assures that the output proof will not share the first 3 rounds with some simulated proof. Hence we do not need to worry about mismatch between the programmed random oracle and the real random oracle. (That is, the fact that $\mathcal{B}$ programmed RO to have challenge $\mathfrak{z}'$ instead of $\mathfrak{z}$ will not be noticed.) Given proof $\pi' = [\mathsf{a}'(\alpha\chi), \mathsf{b}'(\alpha\chi), \mathsf{c}'(\alpha\chi)]_1 \ldots$ for instance x', output by $\mathcal{A}$ adversary $\mathcal{B}$ proceeds as follows
   (a) Translate x' into a corresponding relation in srs: x. More precisely for known w'_i set $\mathsf{w}_i = \mathsf{w}'_i\alpha^i$, $\mathsf{w}_{i+n} = \mathsf{w}'_{i+n}\alpha^i$, $\mathsf{w}_{i+2n} = \mathsf{w}'_{i+2n}\alpha^i$ for $i \in [1 \,..\, \mathsf{n}]$.
   (b) Get commitments $[\mathsf{a}'(\alpha\chi), \mathsf{b}'(\alpha\chi), \mathsf{c}'(\alpha\chi)]_1$ and pass them to the random oracle as $[\mathsf{a}(\chi), \mathsf{b}(\chi), \mathsf{c}(\chi)]_1$, get challenge β, γ.
   (c) Add the commitments and challenges to the proof π.
   (d) Set $\beta' = \beta$ and $\gamma' = \gamma$. Pass the challenges to $\mathcal{A}$.
   (e) Get commitment $[\mathsf{z}'(\alpha\chi)]_1$ and pass it as $[\mathsf{z}(\chi)]_1$, get challenge α.
   (f) Add the commitments and challenges to the proof π.
   (g) Set $\alpha' = \alpha$ and pass it to $\mathcal{A}$.
   (h) Get commitments $\left[\mathsf{t}'_{\mathsf{lo}}(\alpha\chi), \mathsf{t}'_{\mathsf{mid}}(\alpha\chi), \mathsf{t}'_{\mathsf{hi}}(\alpha\chi)\right]_1$ and pass them the random oracle as $[\mathsf{t}_{\mathsf{lo}}(\chi), \mathsf{t}_{\mathsf{mid}}(\chi), \mathsf{t}_{\mathsf{hi}}(\chi)]_1$. Get challenge $\mathfrak{z}$.
   (i) Add the commitments and challenges to the proof π.
   (j) Set $\mathfrak{z}' = \alpha\mathfrak{z}$ and give $\mathfrak{z}'$ to $\mathcal{A}$.
   (k) Get evaluations $\mathsf{a}'(\mathfrak{z}'), \ldots$.
   (l) Add evaluations to the proof π.
   (m) Pass the partial transcript to the random oracle and get challenge ν.
   (n) Set $\nu' = \nu$ and pass it to $\mathcal{A}$.
   (o) Get polynomial openings $\left[W_{\mathfrak{z}'}(\alpha\chi)\right]_1$ and $\left[W_{\mathfrak{z}'\omega}(\alpha\chi)\right]_1$.
   (p) Set $\left[W_{\mathfrak{z}}(\chi)\right]_1 = \left[\alpha W_{\mathfrak{z}'}(\alpha\chi)\right]_1$, and $\left[W_{\mathfrak{z}\omega}(\chi)\right]_1 = \left[\alpha W_{\mathfrak{z}'\omega}(\alpha\chi)\right]_1$.

Since the $\mathcal{A}$'s proof is acceptable, $\mathcal{B}$'s proof is acceptable as well. Hence there is an extractor $\mathsf{Ext}_{\mathcal{B}}$ that outputs witness w given: $\mathcal{B}$, its randomness $r_{\mathcal{B}}$, Q – the list of simulated proofs, $Q_{\mathcal{H}}$ – the list of random oracle responses.

$\mathsf{Ext}_{\mathcal{A}}$ is constructed as follows: $r_{\mathcal{A}} = r_{\mathcal{B}}$, Q' – is a list of simulated proofs, but w.r.t. translations $\mathcal{B}$ made, $Q'_{\mathcal{H}}$ is a list of random oracle responses, but, as in the case of Q', with changes introduced by $\mathcal{B}$.

Michal 20.09: The following is for the setting when there is anhonest SRS at the begining and then the adversary updates it once.

Theorem 1. *Let* $\Psi = (\mathsf{KGen}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *be an* $(2\mu + 1)$*-message* k*-*ur *FSE zkSNARK compiled from an AHP proof system with KZG as the commitment scheme. Let* $i_{\mathfrak{z}}$ *be an index of a round where* P *gets its evaluation challenge* $\mathfrak{z}$*,* $i_{\mathfrak{z}} > k$*. Then* Ψ *is also USE.*

Proof. Let $\mathcal{A}$ be an SE adversary. Since the SNARK is SE, then for $\mathcal{A}$ there exists extractor $\mathsf{Ext}_{\mathcal{A}}$ which takes as input $r_{\mathcal{A}}$ – $\mathcal{A}$'s randomness, Q – list of $\mathcal{A}$ queries to the simulator oracle SimO and their responses, $Q_{\mathcal{H}}$ – list of $\mathcal{A}$'s queries to $\mathcal{H}$ and its responses.

Let $\mathcal{B}$ be a USE adversary. We show how to build extractor $\mathsf{Ext}_{\mathcal{B}}$ using $\mathsf{Ext}_{\mathcal{A}}$. Extractor $\mathsf{Ext}_{\mathcal{B}}$ takes the following input $r_{\mathcal{B}}$ – $\mathcal{B}$'s randomness, Q – list of $\mathcal{B}$'s simulator oracle queries and its responses, $Q_{\mathcal{H}}$

– list of $\mathcal{B}$'s random oracle queries and its responses, Q_{srs} – list of $\mathcal{B}$'s update oracle queries and its responses.

Denote by srs the SRS for $\mathcal{A}$ and by srs' the updated SRS. Let χ be a trapdoor for srs and $\alpha\chi$ be a trapdoor for srs'. As previously, we denote by q the upper bound of the number of random oracle queries $\mathcal{B}$ can make.

Note that a zkSNARK compiled from an AHP consists of the following:

- Commitments to polynomials P sent by the prover; we denote the list of these by C.
- Challenges β_i sent by the verifier; the corresponding list is denoted by B.
- Evaluation challenge $\mathfrak{z}$ sent by the verifier;
- Polynomial evaluations at $\mathfrak{z}$ and proofs of correct evaluations sent by the prover. The list of the former is denoted by E and of the latter by W.

We denote by index i the round when the particular proof element is sent. For example for commitments sent in the second round we write C_2. Here we also additionally require that the set of polynomials sent by the prover in the first round encodes the statement's witness and are masked using vanishing set. Michal 23.09: Formalize it In the following we denote elements sent w.r.t. SRS srs without *apostrophe* and w.r.t srs' *with* an apostrophe. Eventually, $\mathcal{A}$ and $\mathcal{B}$ outputs proofs, we denote by $\mathsf{trans}_{\mathcal{A}}$ and $\mathsf{trans}_{\mathcal{B}}$ partial transcripts of those.

Idea for the proof goes as follows. We define adversary $\mathcal{A}$ that internally runs adversary $\mathcal{B}$. Then we show existance of extractor $\mathsf{Ext}_{\mathcal{B}}$ using existance of $\mathsf{Ext}_{\mathcal{A}}$. $\mathcal{A}$ responds $\mathcal{B}$'s queries it answers as follows:

1. Set $Q_m = \emptyset$, $\mathsf{trans}_{\mathcal{A}} = \emptyset$.
2. Guess indices $I = \{i_1, \ldots, i_k\}$ of random oracle queries used to compute the final proof $\pi_{\mathcal{B}}$. (We allow $\mathcal{A}$ to guess this list adaptively.)
3. If query x has index in I then
   (a) Parse the last message m in x.
      – If $\mathsf{trans}_{\mathcal{B}}$ is a partial transcript for an SRS srs then add m to $\mathsf{trans}_{\mathcal{A}}$
      – If $\mathsf{trans}_{\mathcal{B}}$ is a partial transcript for an SRS srs: (a) if m is the proven instance, then add m/α to $\mathsf{trans}_{\mathcal{A}}$; (b) if $m \in C$, then add m to $\mathsf{trans}_{\mathcal{A}}$; (c) if $m \in E$ then add $m\alpha$; (d) if $m \in W$ then add m/α.
   (b) Compute $y \leftarrow \mathcal{H}(\mathsf{trans}_{\mathcal{A}})$;
   (c) Append y to $\mathsf{trans}_{\mathcal{A}}$.
   (d) Return y.
4. If there is a partial transcript t of a proof in Q_m, such that $t = x$, pick a random element y and set $\mathcal{H}(x) = y$.
5. On other random oracle queries, pass the queries to $\mathcal{H}$ and returns its answer.
6. On $\mathcal{B}$'s request to see simulated proof for $(\mathsf{srs}, \mathsf{x}, \mathsf{w})$, $\mathcal{A}$ passes the query to its SimO and returns its answer π.
7. On $\mathcal{B}$ request to see simulated proof for $(\mathsf{srs}', \mathsf{x}', \mathsf{w}')$, $\mathcal{A}$ does the following:
   (a) $(\mathsf{x}, \mathsf{w}) \leftarrow \mathsf{MoveInstanceBackward}(\mathsf{srs}', \mathsf{x}', \mathsf{w}')$
   (b) Ask SimO for a simulated proof π for (x, w)
   (c) $\pi' \leftarrow \mathsf{MoveProofForward}(\mathsf{srs}, \mathsf{x}, \pi)$
   (d) Return π'
   (e) Add all partial transcripts of (x, π), (x', π') to Q_m
8. On $\mathcal{B}$'s final proof $\pi_{\mathcal{B}}$, $\mathcal{A}$ outputs $\pi_{\mathcal{A}} \leftarrow \mathsf{trans}_{\mathcal{A}}$.

We show that probability that $\mathcal{A}$ fails in outputting an acceptable proof is negligible by a series of games.

Game 0: In this game the adversary $\mathcal{B}$ wins if it outputs an acceptable instance and proof $\mathsf{x}_{\mathcal{B}}, \pi_{\mathcal{B}}$ such that $\mathsf{Ext}_{\mathcal{B}}$ fails to extract the corresponding witness with non-negligible probability. This is a standard USE winning condition for $\mathcal{B}$.

Game 1: In this game the environment aborts if the proof $\pi_{\mathcal{B}}$ adversary outputs utilizes a programmed random oracle response.

Game 0 to Game 1: Note that the adversary $\mathcal{B}$ could output a final proof $\pi_{\mathcal{B}}$ that utilizes a query programmed by $\mathcal{A}$, cf. Item 4. In that case, $\mathcal{A}$ could not return such a proof as it is not valid, because the challenges are computed incorrectly. We argue that probability that $\mathcal{B}$ outputs a valid proof with a programmed RO output is negligible.

Note that the only possibility for $\mathcal{B}$ to utilize a programmed random oracle response while producing an acceptable proof is to have partial proof matching the partial simulated proof. More precisely, if the used programmed challenge β is after m-th message, then $\pi_{\mathcal{B}}[0..m] = \pi_{\mathsf{Sim}}[0..m]$, for some simulated proof π_{Sim}. Importantly, $m < k$, otherwise the adversary breaks k unique response property.

Obviously, if $\pi_{\mathcal{B}}[0..m] = \pi_{\mathsf{Sim}}[0..m]$ then $\pi_{\mathcal{B}}[0..m'] = \pi_{\mathsf{Sim}}[0..m']$ for $m' < m$. We show that it is infeasible to $\mathcal{B}$ to output $\pi_{\mathcal{B}}$ such that $\pi_{\mathcal{B}}[0..1] = \pi_{\mathsf{Sim}}[0..1]$. This is done by showing a reduction to a hiding property of the polynomial commitment scheme. More precisely, let $\mathcal{R}$ be a reduction that utilizes $\mathcal{B}$ to break the hiding property, it proceeds as follows:

1. Get $\mathsf{srs}_{\mathsf{PC}}$ and compute proof system's SRS srs, present the SRS to $\mathcal{B}$.
2. Get instance c – a commitment to unknown polynomial.
3. Set the adversary $\mathcal{A}$ proceedsing as above and oracles: random oracle and simulator oracle for $\mathcal{A}$.
4. For random oracle queries response honestly, i.e. by picking random elements, for simulator oracle response with simulated proof computed using trapdoor-less simulator.
5. Guess which simulated proof π_{Sim} will be utilized in the final proof $\mathcal{B}$ eventually outputs $\pi_{\mathcal{B}}$. Denote π_{Sim} instance by $\mathsf{x}_{\mathsf{Sim}}$.
6. When $\mathcal{A}$ asks SimO for a simulated proof for $\mathsf{x}_{\mathsf{Sim}}$, (i.e. $\mathcal{B}$ asks for $\mathsf{x}_{\mathsf{Sim}}$ w.r.t. SRS srs) or $\mathsf{x}_{\mathsf{Sim}}/\alpha$ (i.e. $\mathcal{B}$ asks for $\mathsf{x}_{\mathsf{Sim}}$ w.r.t. srs' and $\mathcal{A}$ translates it into x/α to make it work with srs) include c to the first message of the simulated proof.
7. Eventually, $\mathcal{B}$ asks for evaluation challenge $\mathfrak{z}$ and outputs a proof $\pi_{\mathcal{B}}$.
8. Since the proof $\mathcal{B}$ output was acceptable, $\mathcal{B}$ correctly evaluated the polynomials sent in the first round. Especially, it evaluated polynomial that commits to c.
9. Rewind $\mathcal{B}$ up to the point challenge $\mathfrak{z}$ is presented and picks another challenge $\mathfrak{z}'$. If for $\mathfrak{z}'$ adversary $\mathcal{B}$ does not output a proof for $\mathsf{x}_{\mathcal{B}}$ – rewind the adversary and pick another evaluation challenge.
10. Eventually, get as many evaluation as necessary to interpolate the polynomial in c. Reveal and return the polynomial.

We note that since the probability that $\mathcal{B}$ outputs a valid proof is non-negligible we can use forking lemma and conclude that with overwhelming probability $\mathcal{R}$ will get necessary number of transcripts in polnomial time, cf. Item 9.

Hence the probability that the adversary wins in Game 0, but not in Game 1 is negligible.

We now show that $\mathcal{B}$ wins Game 1 with negligible probability. More precisely, we show how $\mathsf{Ext}_{\mathcal{B}}$ is constructed. Since Ψ is FSE there exists an extractor $\mathsf{Ext}_{\mathcal{A}}(\mathsf{srs}, r_{\mathcal{A}}, Q_{\mathcal{A}}, Q_{\mathcal{A}}^{\mathcal{H}})$ for $\mathcal{A}$ that returns witness w such that $\mathbf{R}(\mathsf{srs}, \mathsf{x}, \mathsf{w}) = 1$. Extractor $\mathsf{Ext}_{\mathcal{B}}((\mathsf{srs}, \mathsf{srs}'), r_{\mathcal{B}}, Q_{\mathcal{B}}, Q_{\mathcal{B}}^{\mathcal{H}}, Q_{\mathcal{B}}^{\mathsf{srs}})$ proceeds as follows:

- Set $r_{\mathcal{A}} = r_{\mathcal{B}}$,
- Set $Q_{\mathcal{A}} =$
- Set $Q_{\mathcal{A}}^{\mathcal{H}} =$

Probability of guessing I correctly: Rough lower-bound for the probability that $\mathcal{A}$ guesses correctly all indices in I, what is necessary for $\mathcal{A}$ to succeed, is $q^{-\mu}$. Fortunatelly, this bound can be make much tighter. Observe, that if Ψ is k-ur, then $\mathcal{A}$ can tell, after k-th challenge, whether the random oracle query given by $\mathcal{B}$ is for the final proof or not – after k-th challenge $\mathcal{B}$'s messages are already determined by the previous messages and challenges. Hence, probability that $\mathcal{A}$ guesses I correctly is at least q^{-k}.

Definition 13 (Witness-first proof system). *We say that an AHP NIZK proof system Ψ is* witness-first *if*

MoveInstanceBackward$(\mathsf{srs}', \mathsf{x}', \mathsf{w}', \alpha)$

return $(\mathsf{x}, \mathsf{w}) \leftarrow (\mathsf{x}', \mathsf{w}') \cdot \alpha$.

MoveProofForward$(\mathsf{srs}, \mathsf{x}, \pi, \alpha)$	MoveProofBackward$(\mathsf{srs}', \mathsf{x}', \pi', \alpha)$
$\mathsf{x}' \leftarrow \mathsf{x}/\alpha$	$\mathsf{x} \leftarrow \mathsf{x}' \cdot \alpha$
$\pi'.C \leftarrow \pi.C$	$\pi.C \leftarrow \pi'.C$
$\pi'.B \leftarrow \pi.B$	$\pi.B \leftarrow \pi'.B$
$\pi'.\mathfrak{z} \leftarrow \pi.\mathfrak{z}/\alpha$	$\pi.\mathfrak{z} \leftarrow \pi'.\mathfrak{z} \cdot \alpha$
$\pi'.E \leftarrow \pi.E$	$\pi.E \leftarrow \pi'.E$
$\pi'.W \leftarrow \pi.W/\alpha$	$\pi.W \leftarrow \pi'.W \cdot \alpha$
return π'	**return** π

Fig. 3: Moving elements between SRS-s. Michal 23.09: I think that figure may not be needed now.

1. *there are efficient algorithms* Encode, Decode *such that for instance* x*, witness* w*, polynomials* $P = \{\mathsf{p}_1, \ldots, \mathsf{p}_k\}$ *sent by the prover in the first round:*
   - $\mathsf{Encode}(\mathsf{x}, \mathsf{w}) = P$,
   - $\mathsf{Decode}(P) = (\mathsf{x}, \mathsf{w})$.
2. *The proof contains evaluations of each of polynomials in* P *at evaluation challenge* $\mathfrak{z}$.

That is, given polynomials sent by the prover in the first round one can efficiently compute instance's witness.

Lemma 4 (**k-unique response property to 1-unique response property).** *Let* Ψ *be witness-first proof system compiled from AHP* PC *polynomial commitment scheme. Let* P *be a set of polynomials that are committed to by a prover in the first round. Let* $\mathfrak{z}$ *be an evaluation challenge that is given to the prover in Round* k'*, for some* $k' > k$*. Then ...*

Proof.

6 Simulation soundness and forking simulation-extractability—the general result

Equipped with definitional framework of Section 3 we are ready to present the main result of this paper—a proof of simulation soundness and forking simulation extractability of Fiat-Shamir NIZK based on multi-round protocols.

The proofs go 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 2 (Simulation soundness). *Assume that* Ψ *is* k*-programmable HVZK in the standard model, that is* $\varepsilon_{\mathsf{s}}(\lambda)$*-sound and* k-ur *with security* $\varepsilon_{\mathsf{ur}}(\lambda)$*. Then, the probability that a* PPT *adversary* $\mathcal{A}$ *breaks simulation soundness of* Ψ_{FS} *is upper-bounded by* $\varepsilon_{\mathsf{ur}}(\lambda) + q_{\mathcal{H}}^{\mu}\varepsilon_{\mathsf{s}}(\lambda)$*, where* q *is the total number of queries made by the adversary* $\mathcal{A}$ *to a random oracle* $\mathcal{H}\colon \{0,1\}^* \to \{0,1\}^{\lambda}$.

Proof. **Game G_0:** This is a simulation soundness game played between an adversary $\mathcal{A}$ who is given access to a random oracle $\mathcal{H}$ and simulator $\Psi_{\mathsf{FS}}.\mathsf{Sim}$. $\mathcal{A}$ wins if it manages to produce an accepting proof for a false statement. In the following game hops, we upper-bound the probability that this happens.

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

Game 0 to Game 1: We have, $\Pr\left[\mathsf{G}_0 \wedge \overline{\mathsf{Err}_{\mathsf{ur}}}\right] = \Pr\left[\mathsf{G}_1 \wedge \overline{\mathsf{Err}_{\mathsf{ur}}}\right]$ and, from the difference lemma, cf. Lemma 1, $|\Pr[\mathsf{G}_0] - \Pr[\mathsf{G}_1]| \leq \Pr[\mathsf{Err}_{\mathsf{ur}}]$. Thus, to show that the transition from one game to another introduces only minor change in probability of $\mathcal{A}$ winning it should be shown that $\Pr[\mathsf{Err}_{\mathsf{ur}}]$ is small.

We can assume that $\mathcal{A}$ queried the simulator on the instance it wishes to output, i.e. $\mathsf{x}_{\mathcal{A}}$. We show a reduction $\mathcal{R}_{\mathsf{ur}}$ that utilises $\mathcal{A}$ to break the k-ur property of Ψ. Let $\mathcal{R}_{\mathsf{ur}}$ run $\mathcal{A}$ internally as a black-box:

- The reduction answers both queries to the simulator $\Psi_{\mathsf{FS}}.\mathsf{Sim}$ and to the random oracle. It also keeps lists Q, for the simulated proofs, and $Q_{\mathcal{H}}$ for the random oracle queries.
- When $\mathcal{A}$ makes a fake proof $\pi_{\mathcal{A}}$ for $\mathsf{x}_{\mathcal{A}}$, $\mathcal{R}_{\mathsf{ur}}$ looks through lists Q and $Q_{\mathcal{H}}$ until it finds $\pi_{\mathsf{Sim}}[0..k]$ such that $\pi_{\mathcal{A}}[0..k] = \pi_{\mathsf{Sim}}[0..k]$ and a random oracle query $\pi_{\mathsf{Sim}}[k].\mathsf{ch}$ on $\pi_{\mathsf{Sim}}[0..k]$.
- $\mathcal{R}_{\mathsf{ur}}$ returns two proofs for $\mathsf{x}_{\mathcal{A}}$:

$$\pi_1 = (\pi_{\mathsf{Sim}}[1..k], \pi_{\mathsf{Sim}}[k].\mathsf{ch}, \pi_{\mathsf{Sim}}[k+1..\mu+1])$$
$$\pi_2 = (\pi_{\mathsf{Sim}}[1..k], \pi_{\mathsf{Sim}}[k].\mathsf{ch}, \pi_{\mathcal{A}}[k+1..\mu+1])$$

If $\pi_1 = \pi_2$, then $\mathcal{A}$ fails to break simulation soundness, as $\pi_2 \in Q$. On the other hand, if the proofs are not equal, then $\mathcal{R}_{\mathsf{ur}}$ breaks k-ur-ness of Ψ. This happens only with negligible probability $\varepsilon_{\mathsf{ur}}(\lambda)$, hence $\Pr[\mathsf{Err}_{\mathsf{ur}}] \leq \varepsilon_{\mathsf{ur}}(\lambda)$.

Game G_2: This is identical to G_1 except that now the environment aborts if the instance the adversary proves is not in the language.

Game 1 to Game 2: We show that $|\Pr[\mathsf{G}_1] - \Pr[\mathsf{G}_2]| \leq q^{\mu} \cdot \varepsilon_{\mathsf{s}}(\lambda)$, where $\varepsilon_{\mathsf{s}}(\lambda)$ is the probability of breaking soundness of the underlying *interactive* protocol Ψ. Note that $|\Pr[\mathsf{G}_1] - \Pr[\mathsf{G}_2]|$ is the probability that $\mathcal{A}$ outputs an acceptable proof for a false statement which does not break the unique response property (such proofs have been excluded by G_1). Consider a soundness adversary $\mathcal{A}'$ who initiates a proof with Ψ's verifier $\Psi.\mathsf{V}$, internally runs $\mathcal{A}$ and proceeds as follows:

- It guesses indices $i_1, \ldots, i_{\mu}$ such that random oracle queries $h_{i_1}, \ldots, h_{i_{\mu}}$ are the queries used in the $\pi_{\mathcal{A}}$ proof eventually output by $\mathcal{A}$. This is done with probability at least $1/q^{\mu}$ (since there are μ challenges from the verifier in Ψ).
- On input h for the i-th, $i \notin \{i_1, \ldots, i_{\mu}\}$, random oracle query, $\mathcal{A}'$ returns randomly picked y, sets $\mathcal{H}(h) = y$ and stores (h, y) in $Q_{\mathcal{H}}$ if h is sent to $\mathcal{H}$ the first time. If that is not the case, $\mathcal{A}$ finds h in $Q_{\mathcal{H}}$ and returns the corresponding y.
- On input h_{i_j} for the i_j-th, $i_j \in \{i_1, \ldots, i_{\mu}\}$, random oracle query, $\mathcal{A}'$ parses h_{i_j} as a partial proof transcript $\pi_{\mathcal{A}}[1..j]$ and runs Ψ using $\pi_{\mathcal{A}}[j]$ as a $\Psi.\mathsf{P}$'s j-th message to $\Psi.\mathsf{V}$. The verifier responds with a challenge $\pi_{\mathcal{A}}[j].\mathsf{ch}$. $\mathcal{A}'$ sets $\mathcal{H}(h_{i_j}) = \pi_{\mathcal{A}}[j].\mathsf{ch}$. If we guessed the indices correctly we have that $h_{i_{j'}}$, for $j' \leq j$, parsed as $\pi_{\mathcal{A}}[1..j']$ is a prefix of $\pi_{\mathcal{A}}[1..j]$.
- On query $\mathsf{x}_{\mathsf{Sim}}$ to Sim, $\mathcal{A}'$ runs the simulator $\Psi.\mathsf{Sim}$ internally. Note that we require a simulator that only programs the random oracle for $j \geq k$. If the simulator makes a previously unanswered random oracle query with input $\pi_{\mathsf{Sim}}[1..j]$, $1 \leq j < k$, and this is the i_j-th query, it generates $\pi_{\mathsf{Sim}}[j].\mathsf{ch}$ by invoking $\Psi.\mathsf{V}$ on $\pi_{\mathsf{Sim}}[j]$ and programs $\mathcal{H}(h_{i_j}) = \pi_{\mathsf{Sim}}[j].\mathsf{ch}$. It returns π_{Sim}.
- Answers $\Psi.\mathsf{V}$'s final challenge $\pi_{\mathcal{A}}[\mu].\mathsf{ch}$ using the answer given by $\mathcal{A}$, i.e. $\pi_{\mathcal{A}}[\mu]$.

That is, $\mathcal{A}'$ manages to break soundness of Ψ if $\mathcal{A}$ manages to break simulation soundness without breaking the unique response property and $\mathcal{A}'$ correctly guesses the indices of $\mathcal{A}$ random oracle queries. This happens with probability upper-bounded by $|\Pr[\mathsf{G}_1] - \Pr[\mathsf{G}_2]| \cdot 1/q^{\mu}$. Hence $|\Pr[\mathsf{G}_1] - \Pr[\mathsf{G}_2]| \leq q^{\mu} \cdot \varepsilon_{\mathsf{s}}(\lambda)$.

Note that in G_2 the adversary cannot win. Thus the probability that $\mathcal{A}_{\mathsf{ss}}$ is successful is upper-bounded by $\varepsilon_{\mathsf{ur}}(\lambda) + q^{\mu} \cdot \varepsilon_{\mathsf{s}}(\lambda)$. □

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.

Theorem 3 (Forking simulation-extractable multi-message protocols). *Let* $\Psi = (\mathsf{KGen}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *be an interactive* $(2\mu+1)$*-message proof system for* $\mathcal{R}(1^{\lambda})$ *that is honest verifier zero-knowledge in the standard model[***Hamid:*** Is this different from Trapdoor-less simulatable proof system (def. 1)?]*[6]*, 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 a simulation extraction game played between an adversary $\mathcal{A}$ who has given access to a random oracle $\mathcal{H}$ and simulator $\Psi_{\mathsf{FS}}.\mathsf{Sim}$. There is also 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 reveal the corresponding witness. In the following game hops we upper-bound the probability that this happens.

Game G_1**:** This is identical to G_0 except that now the game is aborted if there is a simulated proof π_{Sim} for $\mathsf{x}_{\mathcal{A}}$ such that $(\mathsf{x}_{\mathcal{A}}, \pi_{\mathsf{Sim}}[1..k]) = (\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}[1..k])$. That is, the adversary in its final proof reuses at least k messages from a simulated proof it saw before and the proof is acceptable. Denote that 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 Theorem 2.

Game G_2**:** This is identical to G_1 except that now the environment aborts also when it fails to build a $(1, \ldots, 1, n, 1, \ldots, 1)$-tree of accepting transcripts T by rewinding $\mathcal{A}$. Denote that event by $\mathsf{Err}_{\mathsf{frk}}$.

Game 1 to Game 2: Note that for every acceptable proof $\pi_{\mathcal{A}}$, we may assume that whenever $\mathcal{A}$ outputs in 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[7], 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, $\mathcal{A}$'s code, its randomness r, the output instance $\mathsf{x}_{\mathcal{A}}$ and proof $\pi_{\mathcal{A}}$, as well as the list Q of simulated proofs (and their instances) 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 $y = (\mathsf{srs}, \mathcal{A}, r, \mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}, Q)$ where we set $h_1, \ldots, h_q$ to be the consecutive queries from list $Q_{\mathcal{H}}$. We run $\mathcal{A}$ internally in $\mathcal{Z}$.

To assure that in the first execution of $\mathcal{Z}$ the adversary $\mathcal{A}$ produce the same $(\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}})$ as in the extraction game, $\mathcal{Z}$ provides $\mathcal{A}$ with the same randomness r and answers queries to the random oracle and simulator with pre-recorded responses in $Q_{\mathcal{H}}$ and Q. 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 and simulator, 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 $\mathcal{A}$ 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 $i = 0$.

[6] Crucially, we require that one can provide an indistinguishable simulated proof without any additional knowledge, as e.g knowledge of a SRS trapdoor.

[7] [**?**] calls these queries *fresh*.

Then new random oracle responses are picked for queries indexed by $i, \ldots, q$ 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 $h_i^2, \ldots, h_q^2$. 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 runs internally 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 could reuse a challenge that it learned from observing proofs in Q. In that case, $\mathcal{Z}$ would 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. Denote by $\widetilde{\mathsf{acc}}'$ the probability that algorithm $\mathcal{Z}$, defined in the lemma, produces an accepting proof with a fresh challenge after Round k. Given the discussion above, we can state that $\widetilde{\mathsf{acc}} = \widetilde{\mathsf{acc}}'$.

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

$$\Pr[\mathsf{Err}_{\mathsf{frk}}] \leq 1 - \widetilde{\mathsf{acc}} \cdot \left(\widetilde{\mathsf{acc}}^{n-1}/q^{n-1} + (2^\lambda)!/((2^\lambda - n)! \cdot (2^\lambda)^n) - 1 \right). \tag{6}$$

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 the 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 the correct witness from T, hence 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 probability of $\mathcal{A}$ outputting acceptable 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 $\mathcal{A}$ outputing an acceptable proof as defined in Definition 3. It thus holds

$$\mathsf{ext} \geq \frac{(\mathsf{acc} - \varepsilon_{\mathsf{ur}}(\lambda))^n}{q^{n-1}} - \underbrace{(\mathsf{acc} - \varepsilon_{\mathsf{ur}}(\lambda)) \cdot \left(1 - \frac{(2^\lambda)!}{(2^\lambda - n)! \cdot (2^\lambda)^n}\right) - \varepsilon_{\mathsf{ur}}(\lambda) - \varepsilon_{\mathsf{f}}(\lambda)}_{\varepsilon(\lambda)} . \tag{7}$$

Note that the part of Eq. (7) 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. Thus,

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

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

7 Non-Malleability of P_{FS}

In this section, we show that P_{FS} is simulation-sound and forking simulation-extractable. To that end, we proceed as follows. First, we show that the version of the KZG polynomial commitment scheme that is proposed in the Plonk paper has the unique opening property, cf. Section 2.3 and Lemma 5. This is then used to show that P has the 2-ur property, cf. Lemma 6.

Next, we show that P is forking-sound. That is, given a number of accepting transcripts which match on the first 3 rounds of the protocol we can either recover a correct 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 7.

Given forking-soundness of P, we use the fact that it is also 2-ur and show, in a similar fashion to [**?**], that it is simulation-extractable. That is, we build reductions that given a simulation extractability adversary $\mathcal{A}$ either break the protocol's unique response property or based on forking soundness break the dlog assumption, if extracting a valid witness from a tree of transcripts is impossible. See Corollary 1.

7.1 Unique opening property of PC_P

Lemma 5. *Let PC_P be a batched version of a KZG polynomial commitment, cf. Fig. 5, then PC_P has the unique opening property 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, and $k' \in \mathbb{N}$ 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:

$$\left(\sum_{i=1}^{k} \gamma^{i-1} \cdot \mathsf{f}_i(X) - \sum_{i=1}^{k} \gamma^{i-1} \cdot s_i\right) + r' \left(\sum_{i=1}^{k'} \gamma'^{i-1} \cdot \mathsf{f'}_i(X) - \sum_{i=1}^{k'} \gamma'^{i-1} \cdot s'_i\right) \equiv \mathsf{o}(X)(X - z) + r'\mathsf{o}'(X)(X - z'). \quad (8)$$

Since r' has been picked at random from $\mathbb{F}$, probability that Eq. (8) 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. [**?**]. 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.

Since any discrepancy between the idealised verifier rejection and real verifier acceptance 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 $2\varepsilon_{\mathsf{dlog}} + 1/\mathbb{F}_p$. □

7.2 Unique response property

Lemma 6. *Let 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 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 $\mathcal{A}(\mathbf{R}, \mathsf{srs} = (\left[1, \chi, \ldots, \chi^{\mathsf{n}+2}\right]_1, [\chi]_2))$ be an algebraic adversary tasked to break the 2-ur-ness of 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 is given the SRS and wins if provides two transcripts that match on all 5 messages send 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 $7 \cdot (\varepsilon_{\mathsf{bind}} + 2/|\mathbb{F}_p|)$. 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 $\varepsilon_{\mathsf{bind}} + 2/|\mathbb{F}_p|$.

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 $\varepsilon_{\mathsf{bind}} + 2/|\mathbb{F}_p|$.

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

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

□

7.3 Forking soundness

Lemma 7. *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.*

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 build 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. Eq. (25), 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 $[1, \ldots, \chi^{\mathsf{n}}]_1, [\chi]_2$ and is tasked to output χ.

The reduction $\mathcal{R}_{\mathsf{dlog}}$ proceeds as follows.

1. Build P's SRS srs using the input dlog instance and 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. Return χ.

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. 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 $\mathcal{C}$.
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.
6. Guess which of the simulated polynomials in response s will be used. Let i be the index of this polynomial.
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 $\mathcal{C}$.

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 $\mathcal{C}$. 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 $\mathcal{C}$.

7.4 Honest verifier zero-knowledge

Lemma 8. *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.* (9) *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* $\varepsilon_{\mathsf{zk}}(\lambda)+\varepsilon_{\mathsf{uber}}(\lambda)$.[8]

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 Appendix C.1.

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

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 $b_1,\dots,b_6$ 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. Appendix C.1 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. Appendix A.2 where $\mathsf{R},\mathsf{S},\mathsf{T},\mathsf{F}$ are polynomials over variables $\boldsymbol{B}=B_1,\dots,B_9$ and are defined as follows. Let $E=\{\{2\},\{3,4\},\{5,6\},\{7,8,9\}\}$ and $E'=E\setminus\{2\}$. Let

$$
\begin{aligned}
\mathsf{F}(\boldsymbol{B}) &= \{B_1\}\cup\{B_1B_i \mid i\in A,\ A\in E'\}\cup\{B_1B_iB_j \mid i\in A, j\in B,\ A,B\in E', B\neq A\}\cup\\
&\quad \{B_1B_iB_jB_k \mid i\in A, j\in B, k\in C,\ A,B,C\in E', A\neq B\neq C\neq A\},\\
\mathsf{R}(\boldsymbol{B}) &= \{B_i \mid i\in A,\ A\in E\}\cup\{B_iB_j \mid i\in A, j\in B,\ A\neq B, A,B\in E\}\cup\\
&\quad \{B_iB_jB_k \mid i\in A,\ j\in B,\ k\in C,\ A,B,C \text{ all different and in } E\}\cup\\
&\quad \{B_iB_jB_kB_l \mid i\in A,\ j\in B,\ k\in C,\ l\in D,\ A,B,C,D \text{ all different and in } E\}
\end{aligned}
\tag{9}
$$

[8] 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.

$$\setminus \mathsf{F}(\boldsymbol{B}),$$
$$\mathsf{S}(\boldsymbol{B}) = \emptyset, \qquad \mathsf{T}(\boldsymbol{B}) = \emptyset. \tag{10}$$

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 $\chi, \mathfrak{z}$ 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 $\left[B_iB_j, B_iB_jB_k, B_iB_jB_kB_l\right]_1 \in \mathsf{R}$. This is possible since $\mathcal{R}$ knows all singletons $[w_1, b_2, \ldots, b_9]_1$ and pairs $\left[b_ib_j\right]_1 \in \mathsf{R}$ which can be used to compute all required pairs $\left[\tilde{b}_i\tilde{b}_j\right]_1$:

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

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, $|\Pr[\mathsf{G}_0] - \Pr[\mathsf{G}_2]| \leq \varepsilon_{\mathsf{zk}}(\lambda) + \varepsilon_{\mathsf{uber}}(\lambda)$. □

7.5 Simulation soundness and simulation extractability of P_{FS}

Since Lemmas 6 and 7 hold, P is 2-ur and forking sound. We now make use of Theorem 2 and Theorem 3 and show that P_{FS} is simulation sound and forking simulation-extractable as defined in Section 2.6.

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* $\mathcal{H}\colon \{0,1\}^* \to \{0,1\}^\lambda$ *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 constrains in the proven circuit.*

8 Non-malleability of S_{FS}

8.1 Sonic protocol rolled out

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

The constraint system Sonic's system of constraints composes of three n-long vectors $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ which corresponds to left and right inputs to multiplication gates and their outputs. It hence holds $\boldsymbol{a}\cdot\boldsymbol{b} = \boldsymbol{c}$.

There is also Q linear constrains of the form

$$\boldsymbol{a}\boldsymbol{u}_{\boldsymbol{q}} + \boldsymbol{b}\boldsymbol{v}_{\boldsymbol{q}} + \boldsymbol{c}\boldsymbol{w}_{\boldsymbol{q}} = k_q,$$

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

$$\begin{aligned} \mathsf{u}_{\mathsf{i}}(Y) &= \sum_{q=1}^{\mathsf{Q}} Y^{q+\mathsf{n}} u_{q,i}\,, \qquad & \mathsf{w}_{\mathsf{i}}(Y) &= -Y^i - Y^{-i} + \sum_{q=1}^{\mathsf{Q}} Y^{q+\mathsf{n}} w_{q,i}\,, \\ \mathsf{v}_{\mathsf{i}}(Y) &= \sum_{q=1}^{\mathsf{Q}} Y^{q+\mathsf{n}} v_{q,i}\,, \qquad & \mathsf{k}(Y) &= \sum_{q=1}^{\mathsf{Q}} Y^{q+\mathsf{n}} k_q. \end{aligned} \tag{11}$$

Sonic constraint system requires that

$$\boldsymbol{a}^\top\cdot\mathsf{u}(Y) + \boldsymbol{b}^\top\cdot\mathsf{v}(Y) + \boldsymbol{c}^\top\cdot\mathsf{w}(Y) + \sum_{i=1}^{\mathsf{n}} a_i b_i (Y^i + Y^{-i}) - \mathsf{k}(Y) = 0. \tag{12}$$

In Sonic we will use commitments to the following polynomials.

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

Polynomials $\mathsf{r}(X,Y), \mathsf{s}(X,Y), \mathsf{t}(X,Y)$ are designed such that $\mathsf{t}(0,Y) = \boldsymbol{a}^\top\cdot\mathsf{u}(Y) + \boldsymbol{b}^\top\cdot\mathsf{v}(Y) + \boldsymbol{c}^\top\cdot\mathsf{w}(Y) + \sum_{i=1}^{\mathsf{n}} a_i b_i (Y^i + Y^{-i}) - \mathsf{k}(Y)$. That is, the prover is asked to show that $\mathsf{t}(0,Y) = 0$, cf. Eq. (12).

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

Algorithms rolled out Sonic *SRS generation* $\mathsf{KGen}(\mathbf{R})$. The SRS generating algorithm picks randomly $\alpha, \chi \leftarrow_{\$} \mathbb{F}_p$ and outputs

$$\mathsf{srs} = \left(\left[\{\chi^i\}_{i=-\mathsf{d}}^{\mathsf{d}}, \{\alpha\chi^i\}_{i=-\mathsf{d}, i\neq 0}^{\mathsf{d}}\right]_1, \left[\{\chi^i, \alpha\chi^i\}_{i=-\mathsf{d}}^{\mathsf{d}}\right]_2, [\alpha]_T\right)$$

Sonic *prover* $\mathsf{P}(\mathsf{srs}, \mathsf{x}, \mathsf{w} = \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$.

Round 1 The prover picks randomly randomisers $c_{\mathsf{n}+1}, c_{\mathsf{n}+2}, c_{\mathsf{n}+3}, c_{\mathsf{n}+4} \leftarrow_{\$} \mathbb{F}_p$. Sets $\mathsf{r}(X,Y) \leftarrow \mathsf{r}(X,Y) + \sum_{i=1}^{4} c_{\mathsf{n}+i} X^{-2\mathsf{n}-i}$. Commits to $\mathsf{r}(X,1)$ and outputs $[r]_1 \leftarrow \mathsf{Com}(\mathsf{srs}, \mathsf{n}, \mathsf{r}(X,1))$. Then it gets challenge y from the verifier.

Round 2 P commits to $\mathsf{t}(X,y)$ and outputs $[t]_1 \leftarrow \mathsf{Com}(\mathsf{srs}, \mathsf{d}, \mathsf{t}(X,y))$. Then it gets a challenge z from the verifier.

Round 3 The prover computes commitment openings. That is, it outputs

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

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

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

and $s' = \mathsf{s}(z,y)$. Then it obtains a challenge u from the verifier.

Round 4 In the next round the prover computes $[c]_1 \leftarrow \mathsf{Com}(\mathsf{srs}, \mathsf{d}, \mathsf{s}(u,Y))$ and computes commitments' openings

$$\begin{aligned}[w]_1 &= \mathsf{Op}(\mathsf{srs}, u, \mathsf{s}(u,y), \mathsf{s}(X,y)),\\ \left[q_y\right]_1 &= \mathsf{Op}(\mathsf{srs}, y, \mathsf{s}(u,y), \mathsf{s}(u,Y)),\end{aligned}$$

and returns $[w]_1, \left[q_y\right]_1, s = \mathsf{s}(u,y)$. Eventually the prover gets the last challenge from the verifier—z'.

Round 5 In the final round, P computes opening $[q_{z'}]_1 = \mathsf{Op}(\mathsf{srs}, z', \mathsf{s}(u,z'), \mathsf{s}(u,X))$ and outputs $[q_{z'}]_1$.

Sonic *verifier* $\mathsf{V}(\mathsf{srs}, \mathsf{x}, \pi)$. The verifier in Sonic runs as subroutines the verifier for the polynomial commitment. That is it sets $t' = a'(b' + s') - \mathsf{k}(y)$ and checks the following:

$$\begin{aligned}&\mathsf{PC}_{\mathsf{S}}.\mathsf{V}(\mathsf{srs}, \mathsf{n}, [r]_1, z, a', [o_a]_1),\\ &\mathsf{PC}_{\mathsf{S}}.\mathsf{V}(\mathsf{srs}, \mathsf{n}, [r]_1, yz, b', [o_b]_1),\\ &\mathsf{PC}_{\mathsf{S}}.\mathsf{V}(\mathsf{srs}, \mathsf{d}, [t]_1, z, t', [o_t]_1),\\ &\mathsf{PC}_{\mathsf{S}}.\mathsf{V}(\mathsf{srs}, \mathsf{d}, [s]_1, z, s', [o_s]_1),\end{aligned} \qquad \begin{aligned}&\mathsf{PC}_{\mathsf{S}}.\mathsf{V}(\mathsf{srs}, \mathsf{d}, [s]_1, u, s, [w]_1),\\ &\mathsf{PC}_{\mathsf{S}}.\mathsf{V}(\mathsf{srs}, \mathsf{d}, [c]_1, y, s, \left[q_y\right]_1),\\ &\mathsf{PC}_{\mathsf{S}}.\mathsf{V}(\mathsf{srs}, \mathsf{d}, [c]_1, z', \mathsf{s}(u,z'), [q_{z'}]_1),\end{aligned}$$

and accepts the proof iff all the checks holds. Note that the value $\mathsf{s}(u,z')$ that is recomputed by the verifier uses separate challenges u and z'. This enables the batching of many proof and outsourcing of this part of the proof to an untrusted helper.

8.2 Unique opening property of PC_{S}

Lemma 9. PC_{S} *has the unique opening property in the AGM.*

Proof. Let $z \in \mathbb{F}_p$ be the attribute the polynomial is evaluated at, $[c]_1 \in \mathbb{G}$ be the commitment, $s \in \mathbb{F}_p$ the evaluation value, and $o \in \mathbb{G}$ be the commitment opening. We need to show that for every PPT adversary $\mathcal{A}$ probability

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

is at most negligible.

As noted in [**?**, Lemma 2.2] it is enough to upper bound the probability of the adversary succeeding using the idealised verification equation—which considers equality between polynomials—instead of the real verification equation—which considers equality of the polynomials' evaluations.

For a polynomial f, its degree upper bound max, evaluation point z, evaluation result s, and opening $[o(X)]_1$ the idealised check verifies that

$$\alpha(X^{\mathsf{d}-\mathsf{max}} f(X) \cdot X^{-\mathsf{d}+\mathsf{max}} - s) \equiv \alpha \cdot o(X)(X - z)\,, \tag{13}$$

what is equivalent to

$$f(X) - s \equiv o(X)(X - z)\,. \tag{14}$$

Since $o(X)(X - z) \in \mathbb{F}_p[X]$ then from the uniqueness of polynomial composition, there is only one $o(X)$ that fulfils the equation above. □

8.3 Unique response property

The unique response property of S follows from the unique opening property of the used polynomial commitment scheme PC_{S}.

Lemma 10. *If a polynomial commitment scheme* PC_{S} *is evaluation binding with parameter* $\varepsilon_{\mathsf{bind}}(\lambda)$ *and has unique openings property with parameter* $\varepsilon_{\mathsf{op}}(\lambda)$*, then* S *is* 1-ur *with parameter* $\varepsilon_{\mathsf{ur}}(\lambda) \leq \varepsilon_{\mathsf{bind}}(\lambda) + \varepsilon_{\mathsf{op}}(\lambda)$.

Proof. Let $\mathcal{A}$ be an adversary that breaks 1-ur-ness of S. We consider two cases, depending on which round $\mathcal{A}$ is able to provide at least two different outputs such that the resulting transcripts are acceptable. For the first case we show that $\mathcal{A}$ can be used to break the evaluation binding property of PC_{S}, while for the second case we show that it can be used to break the unique opening property of PC_{S}.

The proof goes similarly to the proof of Lemma 6 thus we provide only draft of it here. In each Round i, for $i > 1$, the prover either commits to some well-defined polynomials (deterministically), evaluates these on randomly picked points, or shows that the evaluations were performed correctly. Obviously, for a committed polynomial p evaluated at point x only one value $y = \mathsf{p}(x)$ is correct. If the adversary was able to provide two different values y and $\tilde{y}$ that would be accepted as an evaluation of p at x then the PC_{S}'s evaluation binding would be broken. Alternatively, if $\mathcal{A}$ was able to provide two openings W and $\tilde{\mathsf{W}}$ for $y = \mathsf{p}(x)$ then the unique opening property would be broken. Hence the probability that $\mathcal{A}$ breaks 1-ur-property of PC_{S} is upper-bounded by $\varepsilon_{\mathsf{bind}}(\lambda) + \varepsilon_{\mathsf{op}}(\lambda)$. □

8.4 Forking soundness

Lemma 11. S *is* $(\varepsilon_{\mathsf{f}}(\lambda), 2, \mathsf{n} + \mathsf{Q} + 1)$*-forking sound against algebraic adversaries who can make up to* S *simulation oracle queries, with*

$$\varepsilon_{\mathsf{s}}(\lambda) \leq \varepsilon_{\mathsf{id}}(\lambda) + \varepsilon_{\mathsf{ldlog}}(\lambda) + 4 \cdot S \cdot \varepsilon_{\mathsf{hid}}(\lambda)\,,$$

where $\varepsilon_{\mathsf{id}}(\lambda)$ *is a soundness error of the idealized verifier,* $\varepsilon_{\mathsf{ldlog}}(\lambda)$ *is security of* (d, d)-ldlog *assumption, and* $\varepsilon_{\mathsf{hid}}(\lambda)$ *is a security of hiding of the polynomial commitment scheme.*

Proof. Similarly as in the case of Plonk, 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 2, the instance x, commitments $[\mathsf{r}(\chi, 1), \mathsf{r}(\chi, y), \mathsf{s}(\chi, y), \mathsf{t}(\chi, y)]_1$, and challenge y are the same. The tree branches after the second round of the protocol where the challenge z is presented, thus tree T is build using different values of 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. Eq. (25), and none of commitments $[\mathsf{r}(\chi, 1), \mathsf{r}(\chi, y), \mathsf{s}(\chi, y), \mathsf{t}(\chi, y)]_1$ 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 S, 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 $(\mathsf{n} + \mathsf{Q} + 1)$ acceptable transcripts for different challenges z. As noted in [**?**] this assures that the correct witness is encoded in $\mathsf{r}(X, Y)$. Hence $\mathsf{Ext}_{\mathsf{tree}}$ can recreate polynomials' coefficients by interpolation and reveal the witness with probability 1. Moreover, 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 $[\mathsf{r}(\chi, 1), \mathsf{r}(\chi, y), \mathsf{s}(\chi, y), \mathsf{t}(\chi, y)]_1$ 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 an instance of a ldlog 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 (d, d)-ldlog instance $\left[\chi^{-\mathsf{d}}, \ldots, \chi^{\mathsf{d}}\right]_1, \left[\chi^{-\mathsf{d}}, \ldots, \chi^{\mathsf{d}}\right]_2$ and is tasked to output χ.

The reduction $\mathcal{R}_{\mathsf{dlog}}$ proceeds as follows.

1. Build S's SRS srs: pick a random α and compute $\left[\alpha\chi^{-\mathsf{d}}, \ldots, \alpha\chi^{-1}, \alpha\chi, \ldots, \alpha\chi^{\mathsf{d}}\right]_1$, $\left[\alpha\chi^{-\mathsf{d}}, \ldots, \alpha\chi^{-1}, \alpha\chi, \ldots, \alpha\chi^{\mathsf{d}}\right]_2$. Compose the 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. Return χ.

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

Game 2: In this game the adversary additionally wins if at least one of the commitments $[\mathsf{r}(\chi, 1), \mathsf{r}(\chi, y), \mathsf{s}(\chi, y)]_1$ utilizes a commitment that comes from a simulated proof; for example, $\mathcal{A}$ could compute its commitment to $\mathsf{r}(X, 1)$ as follows: it picks a polynomial $\mathsf{p}(X, Y)$, computes $[\mathsf{p}(\chi, 1)]_1$, and outputs commitment $[\mathsf{c}(\chi, 1)]_1 = [\mathsf{p}(\chi, 1)]_1 + c$, 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 $[\mathsf{r}(\chi, 1)]_1$.

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. Given polynomial commitment SRS $\mathsf{srs}_{\mathsf{PC_S}}$, produce Sonic's SRS srs similarly to Game 1.
2. Pick random polynomials $\mathsf{p}(X, Y), \mathsf{p}'(X, Y) \in \mathbb{F}^{<|\mathsf{d}|}(X)$, 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.
6. Guess which of the simulated polynomials in response s will be used. Let i be the index of this polynomial.
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 commitment $\mathsf{r}(\chi, 1)$ it gives c.
8. Start $\mathcal{T}'(\mathcal{A}, \mathsf{srs})$ and get the tree T.

9. Since T contains $(\mathsf{n}+\mathsf{Q}+1)$ evaluations of $\mathsf{t}(X,y)$, the polynomial can be reconstructed.
10. Since $\mathcal{A}$ is algebraic, $\mathcal{R}$ learns composition of $\mathsf{t}(X,y)$ in the srs and simulated elements.
11. Hence $\mathcal{R}$ learns whether c is a commitment to $\mathsf{p}(X,1)$ or $\mathsf{p}'(X,1)$.
12. $\mathcal{R}$ returns its guessing bit to C.

Game 3: In this game the adversary additionally wins if the commitment $\mathsf{t}(\chi,y)$ 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.
2. Pick random y, z and compute polynomial $\mathsf{t}(X,Y)$ as a simulator would compute.
3. Produce two random polynomials $\mathsf{p}_0(X,y)$ and $\mathsf{p}_1(X,y)$ and send them to the challenger C. Get commitment c.
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:
   (a) Start making the simulated proof as a trapdoor-less simulator would.
   (b) Before $[\mathsf{t}(z,y)]_1$ is computed get evaluation $p_z = \mathsf{p}_\mathsf{b}(z,y)$, i.e. evaluate the polynomial in c at z.
   (c) Let $\tilde{t}$ be the evaluation of the simulated $\mathsf{t}(z,y)$.
   (d) Pick r such that $p_z + r = \mathsf{t}(z,y)$.
   (e) For the commitment of $\mathsf{t}(X,y)$ output $c' = c + [r]_1$.
   (f) Compute the rest of the simulated proof as a simulator would.
5. 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 $[\mathsf{t}(\chi,y)]_1$ it gives c'.
6. Start $\mathcal{T}'(\mathcal{A},\mathsf{srs})$ and get the tree T.
7. Since T contains $\mathsf{n}+\mathsf{Q}+1$ evaluations of $\mathsf{t}(X,y)$, the polynomial can be reconstructed.
8. Since $\mathcal{A}$ is algebraic, $\mathcal{R}$ learns composition of $\mathsf{t}(X,y)$ in the srs and simulated elements.
9. Hence $\mathcal{R}$ learns whether c is a commitment to $\mathsf{p}(X,y)$ or $\mathsf{p}'(X,y)$.
10. $\mathcal{R}$ returns its guessing bit to C.

8.5 Honest verifier zero-knowledge

Lemma 12. Sonic *is honest verifier zero-knowledge.*

Proof. The simulator proceeds as follows.

1. Pick randomly vectors $\boldsymbol{a}, \boldsymbol{b}$ and set
$$\boldsymbol{c} = \boldsymbol{a} \cdot \boldsymbol{b}. \tag{15}$$
2. Pick randomisers $c_{\mathsf{n}+1}, \ldots, c_{\mathsf{n}+4}$, honestly compute polynomials $\mathsf{r}(X,Y), \mathsf{r}'(X,Y), \mathsf{s}(X,Y)$ and pick randomly challenges y, z.
3. Output commitment $[r]_1 \leftarrow \mathsf{Com}(\mathsf{srs}, \mathsf{n}, \mathsf{r}(X,1))$ and challenge y.
4. Compute
$$\begin{aligned} a' &= \mathsf{r}(z,1), \\ b' &= \mathsf{r}(z,y), \\ s' &= \mathsf{s}(z,y). \end{aligned}$$
5. Pick polynomial $\mathsf{t}(X,Y)$ such that
$$\begin{aligned} &\mathsf{t}(X,y) = \mathsf{r}(X,1)(\mathsf{r}(X,y) + \mathsf{s}(X,y)) - \mathsf{k}(Y) \\ &\mathsf{t}(0,y) = 0 \end{aligned}$$

6. Output commitment $[t]_1 = \mathsf{Com}(\mathsf{srs}, \mathsf{d}, \mathsf{t}(X, y))$ and challenge z.
7. Continue following the protocol.

We note that the simulation is perfect. This comes since, except polynomial $\mathsf{t}(X, Y)$ all polynomials are computed following the protocol. For polynomial $\mathsf{t}(X, Y)$ we observe that in a case of both real and simulated proof the verifier only learns commitment $[t]_1 = \mathsf{t}(\chi, y)$ and evaluation $t' = \mathsf{t}(z, y)$. Since the simulator picks $\mathsf{t}(X, Y)$ such that

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

Values of $[t]_1$ are equal in both proofs. Furthermore, the simulator picks its polynomial such that $\mathsf{t}(0, y) = 0$, hence it does not need the trapdoor to commit to it. (Note that the proof system's SRS does not allow to commit to polynomials which have non-zero constant term). □

Remark 1. As noted in [**?**], Sonic is statistically subversion-zero knowledge (Sub-ZK). As noted in [**?**], one way to achieve subversion zero knowledge is to utilise an extractor that extracts a SRS trapdoor from a SRS-generator. Unfortunately, a NIZK made subversion zero-knowledge by this approach cannot achieve perfect Sub-ZK as one has to count in the probability of extraction failure. However, with the simulation presented in Lemma 12, the trapdoor is not required for the simulator as it is able to simulate the execution of the protocol just by picking appropriate (honest) verifier's challenges. This result transfers to S_{FS}, where the simulator can program the random oracle to provide challenges that fits it.

8.6 From forking soundness and unique response property to forking simulation extractability of S_{FS}

Since Lemmas 10 and 11 hold, S is 1-ur and forking sound. We now make use of Theorem 3 and show that S_{FS} is forking simulation-extractable as defined in Definition 3.

Corollary 2 (Forking simulation extractability of S_{FS}). *Assume that* S *is* 1-ur *with security* $\varepsilon_{\mathsf{ur}}(\lambda) = \varepsilon_{\mathsf{bind}}(\lambda) + \varepsilon_{\mathsf{op}}(\lambda)$ *– where* $\varepsilon_{\mathsf{bind}}(\lambda)$ *is polynomial commitment's binding security,* $\varepsilon_{\mathsf{op}}$ *is polynomial commitment unique opening security – and forking-sound with security* $\varepsilon_{\mathsf{f}}(\lambda)$*. Let* $\mathcal{H} : \{0,1\}^* \to \{0,1\}^\lambda$ *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* S_{FS} *with probability at least* acc*. Then* S_{FS} *is forking simulation-extractable with extraction error* $\eta = \varepsilon_{\mathsf{ur}}(\lambda)$*. The extraction probability* ext *is at least*

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

for some negligible $\varepsilon(\lambda)$*,* n *and* Q *being, respectively, the number of multiplicative and linear constrains of the system.*

9 Non-malleability of M_{FS}

We show that Marlin is forking simulation-extractable. To that end, we show that Marlin has all the required properties: has unique response property, is forking special sound, and its simulator can provide indistinguishable proofs without a trapdoor, just by programming the random oracle.

9.1 Marlin protocol rolled-out

Marlin uses R1CS as arithmetization method. That is, the prover given instance x and witness w and $|\mathsf{H}| \times |\mathsf{H}|$ matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ shows that $\boldsymbol{A}(\mathsf{x}^\top, \mathsf{w}^\top)^\top \circ \boldsymbol{B}(\mathsf{x}^\top, \mathsf{w}^\top)^\top = \boldsymbol{C}(\mathsf{x}^\top, \mathsf{w}^\top)^\top$. (Here $\circ$ is a entry-wise product.)

We assume that the matrices have at most $|\mathsf{K}|$ non-zero entries. Obviously, $|\mathsf{K}| \leq |\mathsf{H}|^2$. Let $b = 3$, the upper-bound of polynomial evaluations the prover has to provide for each of the sent polynomials. Denote by d an upper-bound for $\{|\mathsf{H}| + 2b - 1, 2|\mathsf{H}| + b - 1, 6|\mathsf{K}| - 6\}$.

The idea of showing that the constraint system is fulfilled is as follows. Denote by $\boldsymbol{z} = (\mathsf{x}, \mathsf{w})$. The prover computes polynomials $\mathsf{z}_{\mathsf{A}}(X), \mathsf{z}_{\mathsf{B}}(X), \mathsf{z}_{\mathsf{C}}(X)$ which encode vectors $\boldsymbol{Az}, \boldsymbol{Bz}, \boldsymbol{Cz}$ and have degree

$< |\mathsf{H}|$. Importantly, when constrains are fulfilled, $\mathsf{z}_\mathsf{A}(X)\mathsf{z}_\mathsf{B}(X) - \mathsf{z}_\mathsf{C}(X) = \mathsf{h}_0(X)\mathsf{Z}_\mathsf{H}(X)$, for some $\mathsf{h}_0(X)$ and vanishing polynomial $\mathsf{Z}_\mathsf{H}(X)$. The prover sends commitments to these polynomials and shows that they have been computed correctly. More precisely, it shows that

$$\forall \boldsymbol{M} \in \{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}\}, \forall \kappa \in \mathsf{H}, \mathsf{z}_\mathsf{M}(\kappa) = \sum_{\iota \in \mathsf{H}} \boldsymbol{M}[\kappa, \iota]\mathsf{z}(\iota). \tag{16}$$

The ideal verifier checks the following equalities

$$\begin{aligned} \mathsf{h}_3(\beta_3)\mathsf{Z}_\mathsf{K}(\beta_3) &= \mathsf{a}(\beta_3) - \mathsf{b}(\beta_3)(\beta_3\mathsf{g}_3(\beta_3) + \sigma_3/|\mathsf{K}|) \\ \mathsf{r}(\alpha, \beta_2)\sigma_3 &= \mathsf{h}_2(\beta_2)\mathsf{Z}_\mathsf{H}(\beta_2) + \beta_2\mathsf{g}2(\beta_2) + \sigma_2/|\mathsf{H}| \\ \mathsf{s}(\beta_1) + \mathsf{r}(\alpha, \beta_1)(\sum_{M} \eta_M \mathsf{z}_\mathsf{M}(\beta_1)) - \sigma_2\mathsf{z}(\beta_1) &= \mathsf{h}_1(\beta_1)\mathsf{Z}_\mathsf{H}(\beta_1) + \beta_1\mathsf{g}_1(\beta_1) + \sigma_1/|\mathsf{H}| \\ \mathsf{z}_\mathsf{A}(\beta_1)\mathsf{z}_\mathsf{B}(\beta_1) - \mathsf{z}_\mathsf{C}(\beta_1) &= \mathsf{h}_0(\beta_1)\mathsf{Z}_\mathsf{H}(\beta_1) \end{aligned} \tag{17}$$

where $\mathsf{g}_\mathsf{i}(X), \mathsf{h}_\mathsf{i}(X)$, $i \in [1\,..\,3]$, $\mathsf{a}(X), \mathsf{b}(X), \sigma_1, \sigma_2, \sigma_3$ are polynomials and variables required by the sumcheck protocol which allows verifier to efficiently verify that Eq. (16) holds.

9.2 Unique response property

Lemma 13. *Let* PC *be a commitment of knowledge with security* $\varepsilon_\mathsf{k}(\lambda)$, $\varepsilon_\mathsf{bind}(\lambda)$ *and has unique response property with security* $\varepsilon_\mathsf{op}(\lambda)$. *Then probability that a* PPT *adversary* $\mathcal{A}$ *breaks* M_FS 1-ur *property is at most* $6 \cdot (\varepsilon_\mathsf{bind} + \varepsilon_\mathsf{op} + \varepsilon_\mathsf{k})$ *Michal 8.9: Do we need to add probability that the idealized verifier fails* ε_id*?*

Proof. As in previous proofs, we show the property by game hops. Let $N = \mathsf{g}_1, \mathsf{h}_1, \mathsf{g}_2, \mathsf{h}_2, \mathsf{g}_3, \mathsf{h}_3$. That is, M is a set of all polynomials which commitments are send during the protocol after Round 1.

Game 0: In this game the adversary wins if it breaks evaluation binding, unique opening property, or knowledge soundness of one of commitments for polynomials in N.

Probability that a PPT adversary wins in Game 0, is upper bounded by $6 \cdot (\varepsilon_\mathsf{bind} + \varepsilon_\mathsf{op} + \varepsilon_\mathsf{k})$.

Game 1: In this game the adversary additionally wins if it breaks the 2-ur property of the protocol

Game 0 to Game 1: Probability that the adversary wins in Game 1 but not in Game 0 is 0. This is since the polynomials in N are uniquely determined. W.l.o.g. we analyse probability that adversary is able to produce two (different) pairs of polynomials $(\mathsf{h}_2, \mathsf{g}_2)$ and $(\mathsf{h}_2', \mathsf{g}_2')$ such that

$$\begin{aligned} \mathsf{h}_2(X)\mathsf{Z}_\mathsf{H}(X) + X\mathsf{g}_2(X) &= \mathsf{h}_2(X)\mathsf{Z}_\mathsf{H}(X) + X\mathsf{g}_2(X) \\ (\mathsf{h}_2(X) - \mathsf{h}_2'(X))\mathsf{Z}_\mathsf{H}(X) &= X(\mathsf{g}_2'(X) - \mathsf{g}_2'(X)). \end{aligned}$$

Since $\mathsf{h}_2, \mathsf{g}_2 \in \mathbb{F}^{<|\mathsf{H}|-1}[X]$ and $\mathsf{Z} \in \mathbb{F}^{|\mathsf{H}|}[X]$, LHS has different degree than RHS unless both sides have degree 0. This happens when $\mathsf{h}_2(X) = \mathsf{h}_2'(X)$ and $\mathsf{g}_2(X) - \mathsf{g}_2'(X)$.

9.3 Forking soundness

Lemma 14. *Assume that an idealised* M *verifier fails with probability at most* $\varepsilon_\mathsf{id}(\lambda)$ *and probability that a* PPT *adversary breaks dlog is bounded by* $\varepsilon_\mathsf{dlog}(\lambda)$. *Then* M *is* $(\varepsilon_\mathsf{id}(\lambda) + \varepsilon_\mathsf{dlog}(\lambda) + 7 \cdot S\varepsilon_\mathsf{hid}, 2, d+1)$*-forking sound, where* $\varepsilon_\mathsf{id}(\lambda)$ *is probability that the ideal verifier fails,* $\varepsilon_\mathsf{dlog}(\lambda)$ *is a security of a discrete logarithm,* $\varepsilon_\mathsf{hid}(\lambda)$ *is security of hiding of the polynomial commitment scheme, and* S *is the number of simulator oracle queries made by* $\mathcal{A}$, $S = \mathsf{poly}(\lambda)$.

Proof. The proof goes similarly to the respective proofs for Plonk and Sonic. That is, let srs be M's SRS and denote by srs_1 all SRS's $\mathbb{G}_1$-elements. Let $\mathcal{T}$ be an algebraic adversary that produces a statement x and a $(1, \mathsf{d}+1, 1, 1)$-tree of acceptable transcripts T. Note that in all transcripts the instance x, proof elements $\sigma_1, [\mathsf{w}(\chi), \mathsf{z}_\mathsf{A}(\chi), \mathsf{z}_\mathsf{B}(\chi), \mathsf{z}_\mathsf{C}(\chi), \mathsf{h}_0(\chi), \mathsf{s}(\chi)]_1, [\mathsf{g}_1(\chi), \mathsf{h}_1(\chi)]_1$ and challenges $\alpha, \eta_1, \eta_2, \eta_3$ are common as the transcripts share the first 3 messages. The tree branches after the third message of the protocol where the challenge β_1 is presented, thus tree T is build using different values of β_1.

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. Eq. (17), yet the extractor fails to extract a valid witness out of them.

Probability of $\mathcal{T}$ winning this game is $\varepsilon_{\mathsf{id}}(\lambda)$ as the protocol M, instantiated with the idealised verification equation, is perfectly 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 $(\mathsf{d}+1)$ acceptable transcripts for different challenges β_1, it obtains the same number of different evaluations of polynomials $\mathsf{z}_\mathsf{A}, \mathsf{z}_\mathsf{B}, \mathsf{z}_\mathsf{C}$.

Since the transcripts are acceptable by an idealised verifier, the equality $\mathsf{z}_\mathsf{A}(X)\mathsf{z}_\mathsf{B}(X) - \mathsf{z}_\mathsf{C}(X) = \mathsf{h}_0(X)\mathsf{Z}_\mathsf{H}(X)$ holds and each of z_M, $M \in \{A, B, C\}$, has been computed correctly. Hence, $\mathsf{z}_\mathsf{A}, \mathsf{z}_\mathsf{B}, \mathsf{z}_\mathsf{C}$ encodes the valid witness for the proven statement. Since $\mathsf{z}_\mathsf{A}, \mathsf{z}_\mathsf{B}, \mathsf{z}_\mathsf{C}$ are of degree at most d and there is more than $(\mathsf{d}+1)$ their evaluations known, $\mathsf{Ext}_{\mathsf{tree}}$ can recreate their 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 $\mathsf{ve}_{\mathsf{x},\pi}(X) \neq 0$. That is, 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 adversary 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 $(\mathsf{d}, 1)$-dlog instance $\left[1, \ldots, \chi^{\mathsf{d}}\right]_1, [1, \chi]_2$ and is tasked to output χ. The reduction proceeds as follows—it gives the input instance to the adversary as the SRS. Let $(1, \mathsf{T})$ be the output returned by $\mathcal{A}$. 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 the adversary 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 and $\mathcal{R}_{\mathsf{dlog}}$ can find its zero points. Since $\mathsf{ve}_{\mathsf{x},\pi}(\chi) = 0$, the targeted discrete log value χ is among them. Hence, the probability that this event happens is upper-bounded by $\varepsilon_{\mathsf{dlog}}(\lambda)$.

Game 2: In this game the adversary additionally wins if at least one of the commitments $[\mathsf{z}_A(\chi), \mathsf{z}_B(\chi)]_1$ utilizes a commitment that comes from a simulated proof; for example, $\mathcal{A}$ could compute its commitment to $\mathsf{z}_A(X)$ 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 $[\mathsf{z}_A(\chi)]_1$.

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. Given polynomial commitment SRS $\mathsf{srs}_{\mathsf{PCs}}$, produce Marlin's SRS srs similarly to Game 1.
2. Pick random polynomials $\mathsf{p}(X), \mathsf{p}'(X) \in \mathbb{F}^{<|\mathsf{d}|}(X)$, 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.
6. Guess which of the simulated polynomials in response s will be used. Let i be the index of this polynomial.
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 commitment $\mathsf{z}_A(\chi)$ it gives c.
8. Start $\mathcal{T}'(\mathcal{A}, \mathsf{srs})$ and get the tree T.
9. Since T contains $(\mathsf{d}+1)$ evaluations of $\mathsf{z}_A(X)$, the polynomial can be reconstructed.
10. Since $\mathcal{A}$ is algebraic, $\mathcal{R}$ learns composition of $\mathsf{z}_A(X)$ 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 $\mathcal{C}$.

Game 3: In this game the adversary additionally wins if one of the commitments $[\mathsf{z}_C(\chi), \mathsf{s}(\chi), \mathsf{h}_0(\chi), \mathsf{g}_1(\chi), \mathsf{g}_2(\chi)]_1$ 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. W.l.o.g. we assume that the polynomial that utilizes a simulated element is z_C. $\mathcal{R}$ proceeds as follows:

1. Guess the simulation query index s the polynomial(s) come from.
2. Pick random β_1 and compute polynomial $\mathsf{z}_C(X)$ as a simulator would compute.
3. Produce two random polynomials $\mathsf{p}_0(X)$ and $\mathsf{p}_1(X)$ and send them to the challenger $\mathcal{C}$. Get commitment c.
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:
   (a) Start making the simulated proof as a trapdoor-less simulator would.
   (b) Get evaluation $p_{\beta_1} = \mathsf{p}_\mathsf{b}(\beta_1)$, i.e. evaluate the polynomial in c at β_1.
   (c) Let $\tilde{z_C}$ be the evaluation of the simulated $\mathsf{z}_C(\beta)$.
   (d) Pick r such that $p_{\beta_1} + r = \mathsf{z}_C(\beta)$.
   (e) For the commitment of $\mathsf{z}_C(X)$ output $c' = c + [r]_1$.
   (f) Compute the rest of the simulated proof as a simulator would.
5. 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 $[\mathsf{z}_C(\chi)]_1$ it gives c'.
6. Start $\mathcal{T}'(\mathcal{A}, \mathsf{srs})$ and get the tree T.
7. Since T contains $(\mathsf{d}+1)$ evaluations of $\mathsf{z}_C(X)$, the polynomial can be reconstructed.
8. Since $\mathcal{A}$ is algebraic, $\mathcal{R}$ learns composition of $\mathsf{z}_C(X)$ in the srs and simulated elements.
9. Hence $\mathcal{R}$ learns whether c is a commitment to $\mathsf{p}(X)$ or $\mathsf{p}'(X)$.
10. $\mathcal{R}$ returns its guessing bit to $\mathcal{C}$.

9.4 Honest-verifier zero knowledge

Lemma 15. M *is honest verifier zero-knowledge.*

Proof. The simulator follows the protocol except it picks the challenges $\alpha, \eta_A, \eta_B, \eta_C, \beta_1, \beta_2, \beta_3$ before it picks polynomials it sends.

First, it picks $\tilde{\mathsf{z}}_A(X)$, $\tilde{\mathsf{z}}_B(X)$ at random and $\tilde{\mathsf{z}}_C(X)$ such that $\tilde{\mathsf{z}}_A(\beta_1)\tilde{\mathsf{z}}_B(\beta_1) = \tilde{\mathsf{z}}_C(\beta_1)$. Given the challenges and polynomials $\tilde{\mathsf{z}}_A(X)$, $\tilde{\mathsf{z}}_B(X)$, $\tilde{\mathsf{z}}_C(X)$ the simulator computes $\sigma_1 \leftarrow \sum_{\kappa\in\mathsf{H}} \mathsf{s}(\kappa) + \mathsf{r}(\alpha, X)(\sum_{M\in\{A,B,C\}} \eta_M \tilde{\mathsf{z}}_M(X)) - \sum_{M\in\{A,B,C\}} \eta_M \mathsf{r}_M(\alpha, X)\tilde{\mathsf{z}}(X)$.

Then the simulator starts the protocol and follows it, except it programs the random oracle that on partial transcripts it returns the challenges picked by Sim.

9.5 From forking soundness and unique response property to forking simulation extractability of M_{FS}

Corollary 3. *Assume that* M *is* 1-ur *with security* $\varepsilon_{\mathsf{ur}}(\lambda) = 6\cdot(\varepsilon_{\mathsf{bind}} + \varepsilon_{\mathsf{op}} + \varepsilon_{\mathsf{k}})$*, and forking-sound with security* $\varepsilon_{\mathsf{f}}(\lambda)$*. Let* $\mathcal{H}\colon \{0,1\}^* \to \{0,1\}^\lambda$ *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* M_{FS} *with probability at least* acc*. Then* M_{FS} *is forking simulation-extractable with extraction error* $\eta = \varepsilon_{\mathsf{ur}}(\lambda)$*. The extraction probability* ext *is at least*

$$\mathsf{ext} \geq \frac{1}{q^{\mathsf{d}}}(\mathsf{acc} - \varepsilon_{\mathsf{ur}}(\lambda))^{\mathsf{d}+1} - \varepsilon(\lambda).$$

for some negligible $\varepsilon(\lambda)$*,* d *being, the upper bound of constrains of the system.*

$$\mathsf{ext} \geq q^{-(\mathsf{d}-1)}(\mathsf{acc} - 6\cdot(\varepsilon_{\mathsf{bind}} + \varepsilon_{\mathsf{op}} + \varepsilon_{\mathsf{k}}))^{\mathsf{d}} - \varepsilon(\lambda)\,.$$

10 Further work

We identify a number of problems which we left as further work. First of all, the generalised version of the forking lemma presented in this paper can be generalised even further to include protocols where forking soundness holds for protocols where $\mathsf{Ext}_{\mathsf{tree}}$ extracts a witness from a $(n_1, \ldots, n_\mu)$-tree of acceptable transcripts, where more than one $n_j > 1$. I.e. to include protocols that for witness extraction require transcripts that branch at more than one point.

Although we picked Plonk and Sonic as examples for our framework, it is not limited to SRS-based NIZKs. Thus, it would be interesting to apply it to known so-called transparent zkSNARKs like Bulletproofs [**?**], Aurora [**?**] or AuroraLight [**?**].

Since the rewinding technique and the forking lemma used to show simulation extractability of P_{FS} and S_{FS} come with security loss, it would be interesting to show SE of these protocols directly in the algebraic group model.

Although we focused here only on zkSNARKs, it is worth to investigating other protocols that may benefit from our framework, like e.g. identification schemes.

Last, but not least, this paper would benefit greatly if a more tight version of the generalised forking lemma was provided. However, we have to note here that some of the inequalities used in the proof are already tight, i.e. for specific adversaries, some of the inequalities are already equalities.

A Additional Preliminaries

A.1 Dlog assumptions

Definition 14 (**(q_1, q_2)-dlog assumption).** *Let $\mathcal{A}$ be a* PPT *adversary that gets as input $[1, \chi, \ldots, \chi^{q_1}]_1, [1, \chi, \ldots, \chi^{q_2}]_2$, for some randomly picked $\chi \in \mathbb{F}_p$, then*

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

Definition 15 (**(q_1, q_2)-ldlog assumption).** *Let $\mathcal{A}$ be a* PPT *adversary that gets as input $[\chi^{-q_1}, \ldots, 1, \chi, \ldots, \chi^{q_1}]_1, [\chi^{-q_2}, \ldots, 1, \chi, \ldots, \chi^{q_2}]_2$, for some randomly picked $\chi \in \mathbb{F}_p$, then*

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

A.2 Uber assumption

BBG uber assumption. Also, to be able to show computational honest verifier zero knowledge of Plonk in the standard model, what is required by our reduction, we rely on the *uber assumption* introduced by Boneh et al. [**?**] as presented by Boyen in [**?**].

Let $r, s, t, c \in \mathbb{N} \setminus \{0\}$, Consider vectors of polynomials $\mathsf{R} \in \mathbb{F}_p[X_1, \ldots, X_c]^r$, $\mathsf{S} \in \mathbb{F}_p[X_1, \ldots, X_c]^s$ and $\mathsf{T} \in \mathbb{F}_p[X_1, \ldots, X_c]^t$. Write $\mathsf{R} = (\mathsf{r}_1, \ldots, \mathsf{r}_r)$, $\mathsf{S} = (\mathsf{s}_1, \ldots, \mathsf{s}_s)$ and $\mathsf{T} = (\mathsf{t}_1, \ldots, \mathsf{t}_t)$ for polynomials $\mathsf{r}_i, \mathsf{s}_j, \mathsf{t}_k$.

For a function f and vector $(x_1, \ldots, x_c)$ we write $f(\mathsf{R})$ to denote application of f to each element of R, i.e. $f(\mathsf{R}) = (f(\mathsf{r}_1(x_1, \ldots, x_c), \ldots, f(\mathsf{r}_r(x_1, \ldots, x_c))$. Similarly for applying f to S and T.

Definition 16 (Independence of $\mathsf{R}, \mathsf{S}, \mathsf{T}$). *Let $\mathsf{R}, \mathsf{S}, \mathsf{T}$ be defined as above. We say that polynomial $\mathsf{f} \in \mathbb{F}_p[X_1, \ldots, X_c]$ is* dependent *on $\mathsf{R}, \mathsf{S}, \mathsf{T}$ if there exists $rs + t$ constants $a_{i,j}, b_k$ such that $\mathsf{f} = \sum_{i=1}^{r}\sum_{j=1}^{s} a_{i,j}\mathsf{r}_i\mathsf{s}_j + \sum_{k=1}^{t} b_k\mathsf{t}_k$. We say that f is* independent *if it is not dependent.*

To show (standard-model) zero knowledge of Plonk we utilize a generalization of Boneh-Boyen-Goh's *uber assumption* [**?**] stated as follows (the changed element has been put into a dashbox)

Definition 17 (**$(\mathsf{R}, \mathsf{S}, \mathsf{T}, \mathsf{F}, 1)$-uber assumption).** *Let $\mathsf{R}, \mathsf{S}, \mathsf{T}$ be defined as above, $(x_1, \ldots, x_c, y_1, \ldots, y_d) \leftarrow_\$ \mathbb{F}_p^{c+d}$ and let F be a cardinality-d set of pair-wise independent polynomials which are also independent of $(\mathsf{R}, \mathsf{S}, \mathsf{T})$, cf. Definition 16. Then, for any* PPT *adversary $\mathcal{A}$*

$$\Pr\left[\mathcal{A}([\mathsf{R}(x_1, \ldots x_c)]_1, [\mathsf{S}(x_1, \ldots, x_c)]_2, [\mathsf{T}(x_1, \ldots, x_c)]_T, \boxed{[\mathsf{F}(x_1, \ldots, x_c)]_1}) = 1\right] \approx_\lambda$$
$$\Pr\left[\mathcal{A}([\mathsf{R}(x_1, \ldots x_c)]_1, [\mathsf{S}(x_1, \ldots, x_c)]_2, [\mathsf{T}(x_1, \ldots, x_c)]_T, \boxed{[y_1, \ldots, y_d]_1}) = 1\right].$$

Compared to the original uber assumptions, there are two major changes. First, we require not target group $\mathbb{G}_T$ elements to be indistinguishable, but elements of $\mathbb{G}_1$. Second, Boneh et al.'s assumption works for distinguishers who are given only one challenge polynomial f, i.e. $|\mathsf{F}| = 1$.

We show security of our version of the uber assumption using the generic group model as introduced by Shoup [**?**] where all group elements are represented by random binary strings of length λ. That is, there are random encodings ξ_1, ξ_2, ξ_T which are injective functions from $\mathbb{Z}_p^+$ to $\{0,1\}^\lambda$. We write $\mathbb{G}_i = \{\xi_i(x) \mid x \in \mathbb{Z}_p^+\}$, for $i \in \{1, 2, T\}$. For the sake of clarity we denote by $\xi_{i,j}$ the j-th encoding in group $\mathbb{G}_i$.

Let $\mathsf{P}_i = \{p_1, \ldots, p_{\tau_i}\} \subset \mathbb{F}_p[X_1, \ldots, X_n]$, for $i \in \{1, 2, T\}, \tau_i, n \in \mathbb{N}$, be sets of multivariate polynomials. Denote by $\mathsf{P}_i(x_1, \ldots, x_n)$ a set of evaluations of polynomials in $\mathsf{P_i}$ at $(x_1, \ldots, x_n)$. Denote by $L_i = \{(p_j, \xi_{i,j}) \mid j \leq \tau_i\}$.

Let $\mathcal{A}$ be an algorithm that is given encodings ξ_{i,j_i} of polynomials in P_i for $i \in \{1, 2, T\}, j_i = \tau_i$. There is an oracle O that allows to perform $\mathcal{A}$ the following queries:

Group operations in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$: On input $(\xi_{i,j}, \xi_{i,j'}, i, op)$, $j, j' \leq \tau_i$, $op \in \{\mathtt{add}, \mathtt{sub}\}$, O sets $\tau_i' \leftarrow \tau_i + 1$, computes $p_{i,\tau_i'} = p_{i,j}(x_1, \ldots, x_n) \pm p_{i,j'}(x_1, \ldots, x_n)$ respectively to op. If there is an element $p_{i,k} \in L_i$ such that $p_{i,k} = p_{\tau_i'}$, then the oracle returns encoding of $p_{i,k}$. Otherwise it sets the encoding $\xi_{i,\tau_i'}$ to a new unused random string, adds $(p_{i,\tau_i'}, \xi_{i,\tau_i'})$ to L_i, and returns $\xi_{i,\tau_i'}$.

Bilinear pairing: On input $(\xi_{1,j}, \xi_{2,j'})$ the oracle sets $\tau' \leftarrow \tau_T + 1$ and computes $r_{\tau'} \leftarrow p_{i,j}(x_1, \ldots, x_n) \cdot p_{i,j'}(x_1, \ldots, x_n)$. If $r_{\tau'} \in L_T$ then return encoding found in the list L_T, else pick a new unused random string and set $\xi_{T,\tau'}$ to it. Return the encoding to the algorithm.

Given that, we are ready to show security of our variant of the Boneh et al. uber assumption. The proof goes similarly to the original proof given in [**?**] with minor differences.

Theorem 4 (Security of the uber assumption). *Let $\mathsf{P}_i \in \mathbb{F}_p[X_1, \ldots, X_n]^{m_i}$, for $i \in \{1, 2, T\}$ be τ_i tuples of n-variate polynomials over $\mathbb{F}_p$ and let $\mathsf{F} \in \mathbb{F}_p[X_1, \ldots, X_n]^m$. Let ξ_0, ξ_1, ξ_T, $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be as defined above. If polynomials $f \in \mathsf{F}$ are pair-wise independent and are independent of $\mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_T$, then for any $\mathcal{A}$ that makes up to q queries to the GGM oracle holds:*

$$\left| \Pr\left[\mathcal{A}\begin{pmatrix} \xi_1(\mathsf{P}_1(x_1, \ldots, x_n)), \\ \xi_2(\mathsf{P}_2(x_1, \ldots, x_n)), \\ \xi_T(\mathsf{P}_T(x_1, \ldots, x_n)), \\ \xi_1(\mathsf{F}_0), \xi_1(\mathsf{F}_1) \end{pmatrix} = b \;\middle|\; \begin{matrix} x_1, \ldots, x_n, y_1, \ldots, y_m \leftarrow_\$ \mathbb{F}_p, \\ b \leftarrow_\$ \{0,1\}, \\ \mathsf{F}_b \leftarrow \mathsf{F}(x_1, \ldots, x_n), \\ \mathsf{F}_{1-b} \leftarrow (y_1, \ldots, y_m) \end{matrix} \right] - \frac{1}{2} \right| \leq \frac{d(q + m_1 + m_2 + m_T + m)^2}{2p}$$

Proof. Let C be a challenger that plays with $\mathcal{A}$ in the following game. C maintains three lists

$$L_i = \{(p_j, \xi_{i,j}) \mid j \in [1 \,..\, \tau_i]\},$$

for $i \in \{1, 2, T\}$. Invariant τ states that $\tau_1 + \tau_2 + \tau_T = \tau + m_1 + m_2 + m$.

Challenger C answers $\mathcal{A}$'s oracle queries. However, it does it a bit differently that the oracle O would:

Group operations in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$: On input $(\xi_{i,j}, \xi_{i,j'}, i, op)$, $j, j' \leq \tau_i$, $op \in \{\mathtt{add}, \mathtt{sub}\}$, C sets $\tau' \leftarrow \tau_i + 1$, computes $p_{i,\tau'}(X_1, \ldots, X_n) = p_{i,j}(X_1, \ldots, X_n) \pm p_{i,j'}(X_1, \ldots, X_n)$ respectively to op. If there is a polynomial $p_{i,k}(X_1, \ldots, X_n) \in L_i$ such that $p_{i,k}(X_1, \ldots, X_n) = p_{\tau'}(X_1, \ldots, X_n)$, then the challenger returns encoding of $p_{i,k}$. Otherwise it sets the encoding $\xi_{i,\tau'}$ to a new unused random string, adds $(p_{i,\tau'}, \xi_{i,\tau'})$ to L_i, and returns $\xi_{i,\tau'}$.

Bilinear pairing: On input $(\xi_{1,j}, \xi_{2,j'})$ the challenger sets $\tau' \leftarrow \tau_T + 1$ and computes $r_{\tau'}(X_1, \ldots, X_n) \leftarrow p_{i,j}(X_1, \ldots, X_n) \cdot p_{i,j'}(X_1, \ldots, X_n)$. If $r_{\tau'}(X_1, \ldots, X_n) \in L_T$, C returns encoding found in the list L_T. Else it picks a new unused random string and set $\xi_{T,\tau'}$ to it. Finally it returns the encoding to the algorithm.

After at most q queries to the oracle, the adversary returns a bit b'. At that point the challenger C chooses randomly $x_1, \ldots, x_n, y_1 \ldots, y_m$, random bit b, and sets $X_i = x_i$, for $i \in [1 \,..\, n]$, and $Y_i = y_i$, for $i \in [1 \,..\, m]$; furthermore, $\mathsf{F}_b \leftarrow \mathsf{F}(x_1, \ldots, x_n)$ and $\mathsf{F}_{1-b} \leftarrow (y_1, \ldots, y_m)$. Note that C simulates perfectly unless the chosen values $x_1, \ldots, x_n, y_1, \ldots, y_m$ result in equalities between polynomial evaluations that are not equalities between the polynomials. That is, the simulation is perfect unless for some i, j, j' holds

$$p_{i,j}(x_1, \ldots, x_n) - p_{i,j'}(x_1, \ldots, x_n) = 0,$$

for $p_{i,j}(X_1, \ldots, X_n) \neq p_{i,j'}(X_1, \ldots, X_n)$. Denote by bad an event that at least one of the three conditions holds. When bad happens, the answer C gives to $\mathcal{A}$ differs from an answer that a real oracle would give. We bound the probability that bad occurs in two steps.

First we set $\mathsf{F}_b = \mathsf{F}(X_1, \ldots, X_n)$. Note that symbolic substitutions do not introduce any new equalities in $\mathbb{G}_1$. That is, if for all j, j' holds $p_{1,j} \neq p_{1,j'}$, then $p_{1,j} \neq p_{1,j'}$ even after setting $\mathsf{F}_b = \mathsf{F}(X_1, \ldots, X_n)$. This follows since all polynomials in F are pairwise independent and F independent on $\mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_T$. Indeed, $p_{1,j} - p_{1,j'}$ is a polynomial of the form

$$\sum_{j=1}^{m_1} a_j p_{1,j} + \sum_{j=1}^{m} b_j f_j(X_1, \ldots, X_n),$$

for some constants a_j, b_j. If the polynomial is non-zero, but setting $\mathsf{F}_b = \mathsf{F}(X_1, \ldots, X_n)$ makes this polynomial vanish, then some f_k must be dependent on some $\mathsf{P}_1, \mathsf{F} \setminus \{f_k\}$.

Now we set $X_1 \ldots, X_n, \mathsf{F}_{1-b}$ and bound probability that for some i and j, j' holds $(p_{i,j}(x_1, \ldots, x_n) - p_{i,j'}(x_1, \ldots, x_n) = 0$ for $p_{i,j} \neq p_{i,j'}$. By the construction, the maximum total degree of these polynomials is $d = \max(d_{\mathsf{P}_1} + d_{\mathsf{P}_2}, d_{\mathsf{P}_T}, d_{\mathsf{F}})$, where d_f is the total degree of some polynomial f and for a set of polynomials $F = \{f_1, \ldots, f_k\}$, we write $d_F = \{d_{f_1}, \ldots, d_{f_k}\}$. Thus, for a given j, j' probability that a random assignment to $X_1, \ldots, X_n, Y_1, \ldots, Y_n$ is a root of $p_{i,j} - p_{i,j'}$ is, by the Schwartz-Zippel lemma, bounded by d/p, which is negligible. There is at most $2 \cdot \binom{q+m_0+m_1+m}{2}$ such pairs $p_{i,j}, p_{i,j'}$ we have that

$$\Pr[\mathsf{bad}] \leq \binom{q + m_0 + m_1 + m}{2} \cdot \frac{2d}{p} \leq (q + m_0 + m_1 + m)^2 \frac{d}{p}.$$

As noted, if bad does not occur then the simulation is perfect. Also the bit b has been chosen independently on the $\mathcal{A}$'s view, thus $\Pr[b = b' \mid \neg\mathsf{bad}] = 1/2$. Hence,

$$\Pr[b = b'] \leq \Pr[b = b' \mid \neg\mathsf{bad}](1 - \Pr[\mathsf{bad}]) + \Pr[\mathsf{bad}] = \frac{1}{2} + \frac{\Pr[\mathsf{bad}]}{2}$$
$$\Pr[b = b'] \geq \Pr[b = b' \mid \neq \mathsf{bad}](1 - \Pr[\mathsf{bad}]) = \frac{1}{2} - \frac{\Pr[\mathsf{bad}]}{2}.$$

Finally,

$$\left|\Pr[b = b'] - \frac{1}{2}\right| \leq \Pr[\mathsf{bad}]/2 \leq (q + m_0 + m_1 + m)^2 \frac{d}{2p}$$

as required.

A.3 Special simulation-extractability of sigma protocols and forking lemma

Theorem 5 (Special simulation extractability of the Fiat–Shamir transform [?]). *Let* $\Sigma = (\mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *be a non-trivial sigma protocol with unique responses for a language* $\mathcal{L} \in \mathsf{NP}$. *In the random oracle model, the NIZK proof system* $\Sigma_{\mathsf{FS}} = (\mathsf{P}_{\mathsf{FS}}, \mathsf{V}_{\mathsf{FS}}, \mathsf{Sim}_{\mathsf{FS}})$ *resulting by applying the Fiat–Shamir transform to* Σ *is special simulation extractable with extraction error* $\eta = q/h$ *for the simulator* Sim. *Here,* q *is the number of random oracle queries and* h *is the number of elements in the range of* $\mathcal{H}$.

The theorem relies on the following *general forking lemma* [**?**].

Lemma 16 (General forking lemma, cf. [?, ?]). *Fix $q \in \mathbb{Z}$ and a set H of size $h > 2$. Let $\mathcal{Z}$ be a* PPT *algorithm that on input $y, h_1, \ldots, h_q$ returns (i, s), where $i \in [0 \,..\, q]$ and s is called a* side output. *Denote by* IG *a randomised instance generator. We denote by* acc *the probability*

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

Let $\mathsf{F}_{\mathcal{Z}}(y)$ denote the algorithm described in Fig. 4, then the probability frk *defined as* $\mathsf{frk} := \Pr[b = 1 \mid y \leftarrow \mathsf{IG}; (b, s, s') \leftarrow \mathsf{F}_{\mathcal{Z}}(y)]$ *holds*

$$\mathsf{frk} \geq \mathsf{acc}\left(\frac{\mathsf{acc}}{q} - \frac{1}{h}\right) .$$

$\mathsf{F}_{\mathcal{Z}}(y)$

$\rho \leftarrow\!\!\$\, \mathsf{R}(\mathcal{Z})$
$h_1, \ldots, h_q \leftarrow\!\!\$\, H$
$(i, s) \leftarrow \mathcal{Z}(y, h_1, \ldots, h_q; \rho)$
if $i = 0$ **return** $(0, \perp, \perp)$
$h'_i, \ldots, h'_q \leftarrow\!\!\$\, H$
$(i', s') \leftarrow \mathcal{Z}(y, h_1, \ldots, h_{i-1}, h'_i, \ldots, h'_q; \rho)$
if $(i = i') \wedge (h_i \neq h'_i)$ **return** $(1, s, s')$
  else return $(0, \perp, \perp)$

Fig. 4: Forking algorithm $\mathsf{F}_{\mathcal{Z}}$

A.4 Proof of the generalized forking lemma (Lemma 2)

Proof. First denote by $\mathsf{acc}(y)$ and $\mathsf{frk}(y)$ the following probabilities

$$\mathsf{acc}(y) = \Pr\left[i \neq 0 \;\middle|\; h_1, \ldots, h_q \leftarrow\!\!\$\, H;\ (i, s) \leftarrow \mathcal{Z}(y, h_1, \ldots, h_q)\right] .$$
$$\mathsf{frk}(y) = \Pr\left[b = 1 \;\middle|\; (b, \boldsymbol{s}) \leftarrow \mathsf{GF}^m_{\mathcal{Z}}(y, h_1, \ldots, h_q)\right] .$$

We start by claiming that for all y

$$\mathsf{frk}(y) \geq \frac{\mathsf{acc}(y)^m}{q^{m-1}} - \mathsf{acc}(y) \cdot \left(1 - \frac{h!}{(h-m)! \cdot h^m}\right) \tag{18}$$

Then with the expectation taken over $y \leftarrow\!\!\$\, \mathsf{IG}$, we have

$$\mathsf{frk} = \mathbb{E}\left[\mathsf{frk}(y)\right] \geq \mathbb{E}\left[\frac{\mathsf{acc}(y)^m}{q^{m-1}} - \mathsf{acc}(y) \cdot \left(1 - \frac{h!}{(h-m)! \cdot h^m}\right)\right] \tag{19}$$
$$\geq \frac{\mathbb{E}\left[\mathsf{acc}(y)\right]^m}{q^{m-1}} - \mathbb{E}\left[\mathsf{acc}(y)\right] \cdot \left(1 - \frac{h!}{(h-m)! \cdot h^m}\right) \tag{20}$$
$$= \frac{\mathsf{acc}^m}{q^{m-1}} - \mathsf{acc} \cdot \left(1 - \frac{h!}{(h-m)! \cdot h^m}\right) . \tag{21}$$

Where Eq. (19) comes from Eq. (18); Eq. (20) comes from linearity of expected value and Lemma 17; and Eq. (21) holds by the fact that $\mathbb{E}\left[\mathsf{acc}(y)\right] = \mathsf{acc}$.

We now show Eq. (18). Denote by $J = [1 \,..\, m]^2 \setminus \{(j, j)\}_{j \in [1 \,..\, m]}$. For any input y, with probabilities taken over the coin tosses of $\mathsf{GF}^m_{\mathcal{Z}}$ we have

$$\mathsf{frk}(y) = \Pr\left[i_j = i_{j'} \wedge i_j \geq 1 \wedge h^j_{i_j} \neq h^{j'}_{i_{j'}} \text{ for } (j, j') \in J\right]$$

$$\geq \Pr\left[i_j = i_{j'} \wedge i_j \geq 1 \text{ for } (j,j') \in J\right] - \Pr\left[i_j \geq 1 \wedge h_{i_j}^{j} = h_{i_{j'}}^{j'} \text{ for some } (j,j') \in J\right]$$
$$= \Pr\left[i_j = i_{j'} \wedge i_j \geq 1 \text{ for } (j,j') \in J\right] - \Pr\left[i_j \geq 1\right] \cdot \left(1 - \frac{h!}{(h-m)! \cdot h^m}\right)$$
$$= \Pr\left[i_j = i_{j'} \wedge i_j \geq 1 \text{ for } (j,j') \in J\right] - \mathsf{acc}(y) \cdot \left(1 - \frac{h!}{(h-m)! \cdot h^m}\right).$$

Probability that for some $(j,j') \in J$ and $i_j = i_{j'}$ holds $h_{i_j}^{j} \neq h_{i_{j'}}^{j'}$ equals

$$\frac{h \cdot (h-1) \cdot \ldots \cdot (h-m-1)}{h^m} = \frac{h!}{(h-m)! \cdot h^m}.$$

That is, it equals the number of all m-element strings where each element is different divided by the number of all m-element strings, where elements are taken from a set of size h.

It remains to show that $\Pr\left[i_j = i_{j'} \wedge i_j \geq 1 \text{ for } (j,j') \in J\right] \geq \mathsf{acc}(y)^m / q^{m-1}$. Let $\mathsf{R}(\mathcal{Z})$ denote the set from which $\mathcal{Z}$ picks its coins at random. For each $\iota \in [1\,..\,q]$ let $X_\iota \colon \mathsf{R}(\mathcal{Z}) \times H^{\iota-1} \to [0,1]$ be defined by setting $X_\iota(\rho, h_1, \ldots, h_{\iota-1})$ to

$$\Pr\left[i = \iota \;\middle|\; h_\iota, \ldots, h_q \leftarrow\!\!\$\, H; (i,s) \leftarrow \mathcal{Z}(y, h_1, \ldots, h_q; \rho)\right]$$

for all $\rho \in \mathsf{R}(\mathcal{Z})$ and $h_1, \ldots, h_{\iota-1} \in H$. Consider X_ι be a random variable over the uniform distribution on its domain. Then

$$\Pr\left[i_j = i_{j'} \wedge i_j \geq 1 \text{ for } (j,j') \in J\right] = \sum_{\iota=1}^{q} \Pr\left[i_1 = \iota \wedge \ldots \wedge i_m = \iota\right]$$
$$= \sum_{\iota=1}^{q} \Pr\left[i_1 = \iota\right] \cdot \Pr\left[i_2 = \iota \mid i_1 = \iota\right] \cdot \ldots \cdot \Pr\left[i_m = \iota \mid i_1 = \ldots = i_{m-1} = \iota\right]$$
$$= \sum_{\iota=1}^{q} \sum_{\rho, h_1, \ldots, h_{\iota-1}} X_\iota(\rho, h_1, \ldots, h_{\iota-1})^m \cdot \frac{1}{|\mathsf{R}(\mathcal{Z})| \cdot |H|^{\iota-1}} = \sum_{\iota=1}^{q} \mathbb{E}\left[X_\iota^m\right].$$

Importantly, $\sum_{\iota=1}^{q} \mathbb{E}\left[X_\iota\right] = \mathsf{acc}(y)$.

By Lemma 17 we get

$$\sum_{\iota=1}^{q} \mathbb{E}\left[X_\iota^m\right] \geq \sum_{\iota=1}^{q} \mathbb{E}\left[X_\iota\right]^m.$$

Note that for e.g. $X_i = 1$, $i \in [1\,..\,q]$ the inequality becomes equality, that is, it is tight.

We now use the Hölder inequality, cf. Lemma 18, for $x_i = \mathbb{E}\left[X_i\right]$, $y_i = 1$, $p = m$, and $q = m/(m-1)$ obtaining

$$\left(\sum_{i=1}^{q} \mathbb{E}\left[X_i\right]\right)^m \leq \left(\sum_{i=1}^{q} \mathbb{E}\left[X_i\right]^m\right) \cdot q^{m-1} \tag{22}$$

$$\frac{1}{q^{m-1}} \cdot \mathsf{acc}(y)^m \leq \sum_{i=1}^{q} \mathbb{E}\left[X_i\right]^m. \tag{23}$$

Finally, we get

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

□

Lemma 17. *Let $\mathsf{R}(\mathcal{Z})$ denote the set from which $\mathcal{Z}$ picks its coins at random. For each $\iota \in [1\,..\,q]$ let $X_\iota \colon \mathsf{R}(\mathcal{Z}) \times H^{\iota-1} \to [0,1]$ be defined by setting $X_\iota(\rho, h_1, \ldots, h_{\iota-1})$ to*

$$\Pr\left[i = \iota \;\middle|\; h_\iota, \ldots, h_q \leftarrow\!\!\$\, H; (i,s) \leftarrow \mathcal{Z}(y, h_1, \ldots, h_q; \rho)\right]$$

for all $\rho \in \mathsf{R}(\mathcal{Z})$ and $h_1, \ldots, h_{\iota-1} \in H$. Consider X_ι as a random variable over the uniform distribution on its domain. Then $\mathbb{E}\left[X_\iota^m\right] \geq \mathbb{E}\left[X_\iota\right]^m$.

Proof. First we recall the Jensen inequality [**?**], if for some random variable X holds $|\mathbb{E}[X]| \leq \infty$ and f is a Borel convex function then

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)] \ .$$

Finally, we note that $|\mathbb{E}[X]| \leq \infty$ and taking to the m-th power is a Borel convex function on $[0, 1]$ interval. □

Lemma 18 (Hölder's inequality. Simplified.). *Let x_i, y_i, for $i \in [1 \,..\, q]$, and p, q be real numbers such that $1/p + 1/q = 1$. Then*

$$\sum_{i=1}^{q} x_i y_i \leq \left(\sum_{i=1}^{q} x_i^p\right)^{\frac{1}{p}} \cdot \left(\sum_{i=1}^{q} y_i^p\right)^{\frac{1}{q}} \ . \tag{24}$$

Remark 2 (Tightness of the Hölder inequality). In is important to note that Inequality (24) is tight. More precisely, for $\mathbb{E}[X_i] = x$, $i \in [1 \,..\, q]$ we have

$$\sum_{i=1}^{q} x = \left(\sum_{i=1}^{q} x^m\right)^{\frac{1}{m}} \cdot \left(\sum_{i=1}^{q} 1^{\frac{m}{m-1}}\right)^{\frac{m-1}{m}}$$
$$qx = (qx^m)^{\frac{1}{m}} \cdot q^{\frac{m-1}{m}}$$
$$(qx)^m = qx^m \cdot q^{m-1}$$
$$(qx)^m = (qx)^m \ .$$

Lemma 19. *Let $\mathsf{f}(X)$ be a random degree-d polynomial over $\mathbb{F}_p[X]$. Then the probability that $\mathsf{f}(X)$ has roots in $\mathbb{F}_p$ is at least $1/d!$.*

Proof. First observe that there is p^d canonical polynomials in $\mathbb{F}_p[X]$. Each of the polynomials may have up to d roots. Consider polynomials which are reducible to polynomials of degree 1, i.e. polynomials that have all d roots. The roots can be picked in $\bar{C}_d^p$ ways, where $\bar{C}_k^n$ is the number of k-elements combinations with repetitions from n-element set. That is,

$$\bar{C}_k^n = \binom{n+k-1}{k} .$$

Thus, the probability that a randomly picked polynomial has all d roots is

$$p^{-d} \cdot \bar{C}_d^p = p^{-d} \cdot \binom{p+d-1}{d} = p^{-d} \cdot \frac{(p+d-1)!}{(p+d-1-d)! \cdot d!} =$$
$$p^{-d} \cdot \frac{(p+d-1) \cdot \ldots \cdot p \cdot (p-1)!}{(p-1)! \cdot d!} = p^{-d} \cdot \frac{(p+d-1) \cdot \ldots \cdot p}{d!}$$
$$\geq p^{-d} \cdot \frac{p^d}{d!} = \frac{1}{d!}$$

□

B Omitted protocols descriptions

B.1 Polynomial commitment schemes

Figs. 5 and 6 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.

We emphasize the following properties of a secure polynomial commitment PC:

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 $\mathcal{A}$

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

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 $\mathcal{A}$

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

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 5 and Lemma 9 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

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

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

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

In that case we say that 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. [**?**] 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 $i \in [1 \,..\, |\boldsymbol{z}|]$ **do**

$\quad \mathsf{o}_i(X) \leftarrow \sum_{j=1}^{t_i} \gamma_i^{j-1} \frac{\mathsf{f}_{i,j}(X) - \mathsf{f}_{i,j}(z_i)}{X - z_i}$

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

$\quad$ **if** $\sum_{i=1}^{|\boldsymbol{z}|} r_i \cdot \left[\sum_{j=1}^{t_j} \gamma_i^{j-1} c_{i,j} - \sum j = 1^{t_j} s_{i,j}\right]_1 \bullet [1]_2 +$

$\quad \sum_{i=1}^{|\boldsymbol{z}|} r_i z_i o_i \bullet [1]_2 \neq \left[-\sum_{i=1}^{|\boldsymbol{z}|} r_i o_i\right]_1 \bullet [\chi]_2$ **then**

$\quad\quad$ **return** 0

return 1.

Fig. 5: PC_P polynomial commitment scheme.

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

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

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

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

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

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

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

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

else return 0.

Fig. 6: PC_S polynomial commitment scheme.

C Non-malleability of Plonk, omitted proofs and descriptions

C.1 Plonk protocol rolled out

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

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

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

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

Algorithms rolled out Plonk argument system is universal. That is, it allows to verify computation of any arithmetic circuit which has no more than n gates using a single SRS. However, to make computation efficient, for each circuit there is allowed a preprocessing phase which extend the SRS with circuit-related polynomial evaluations.

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

Plonk *SRS generating algorithm* $\mathsf{KGen}(\mathbf{R})$*:* The SRS generating algorithm picks at random $\chi \leftarrow\!\!\$\, \mathbb{F}_p$, computes and outputs

$$\mathsf{srs} = \left(\left[\{\chi^i\}_{i=0}^{\mathsf{n}+2} \right]_1, [\chi]_2 \right).$$

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

$$\mathsf{S}_{\mathsf{id}1}(X) = X, \qquad \mathsf{S}_{\sigma 1}(X) = \sum_{i=1}^{\mathsf{n}} \sigma(i) \mathsf{L}_i(X),$$

$$\mathsf{S}_{\mathsf{id}2}(X) = k_1 \cdot X, \qquad \mathsf{S}_{\sigma 2}(X) = \sum_{i=1}^{\mathsf{n}} \sigma(\mathsf{n}+i) \mathsf{L}_i(X),$$

$$\mathsf{S}_{\mathsf{id}3}(X) = k_2 \cdot X, \qquad \mathsf{S}_{\sigma 3}(X) = \sum_{i=1}^{\mathsf{n}} \sigma(2\mathsf{n}+i) \mathsf{L}_i(X).$$

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

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

$$\mathsf{q}_{\mathsf{L}}(X) = \sum_{i=1}^{\mathsf{n}} \boldsymbol{q_L}_i \mathsf{L}_i(X), \qquad \mathsf{q}_{\mathsf{O}}(X) = \sum_{i=1}^{\mathsf{n}} \boldsymbol{q_O}_i \mathsf{L}_i(X),$$

$$\mathsf{q}_{\mathsf{R}}(X) = \sum_{i=1}^{\mathsf{n}} \boldsymbol{q_R}_i \mathsf{L}_i(X), \qquad \mathsf{q}_{\mathsf{C}}(X) = \sum_{i=1}^{\mathsf{n}} \boldsymbol{q_C}_i \mathsf{L}_i(X).$$

$$\mathsf{q}_{\mathsf{M}}(X) = \sum_{i=1}^{\mathsf{n}} \boldsymbol{q_M}_i \mathsf{L}_i(X),$$

Plonk *prover* $\mathsf{P}(\mathsf{srs}, \mathsf{x}, \mathsf{w} = (\mathsf{w}_i)_{i \in [1 \,..\, 3 \cdot \mathsf{n}]})$.

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

$$\mathsf{a}(X) = (b_1 X + b_2)\mathsf{Z_H}(X) + \sum_{i=1}^{\mathsf{n}} \mathsf{w}_i \mathsf{L}_i(X)$$

$$\mathsf{b}(X) = (b_3 X + b_4)\mathsf{Z_H}(X) + \sum_{i=1}^{\mathsf{n}} \mathsf{w}_{\mathsf{n}+i} \mathsf{L}_i(X)$$

$$\mathsf{c}(X) = (b_5 X + b_6)\mathsf{Z_H}(X) + \sum_{i=1}^{\mathsf{n}} \mathsf{w}_{2\cdot\mathsf{n}+i} \mathsf{L}_i(X)$$

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

Round 2 Get challenges $\beta, \gamma \in \mathbb{F}_p$

$$\beta = \mathcal{H}(\pi[0..1], 0)\,, \qquad \gamma = \mathcal{H}(\pi[0..1], 1)\,.$$

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

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

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

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

$$\begin{aligned} &\mathsf{t}(X) = \\ &(\mathsf{a}(X)\mathsf{b}(X)\mathsf{q_M}(X) + \mathsf{a}(X)\mathsf{q_L}(X) + \mathsf{b}(X)\mathsf{q_R}(X) + \mathsf{c}(X)\mathsf{q_O}(X) + \mathsf{PI}(X) + \mathsf{q_C}(X))\frac{1}{\mathsf{Z_H}(X)} + \\ &+ ((\mathsf{a}(X) + \beta X + \gamma)(\mathsf{b}(X) + \beta k_1 X + \gamma)(\mathsf{c}(X) + \beta k_2 X + \gamma)\mathsf{z}(X))\frac{\alpha}{\mathsf{Z_H}(X)} \\ &- (\mathsf{a}(X) + \beta \mathsf{S}_{\sigma 1}(X) + \gamma)(\mathsf{b}(X) + \beta \mathsf{S}_{\sigma 2}(X) + \gamma)(\mathsf{c}(X) + \beta \mathsf{S}_{\sigma 3}(X) + \gamma)\mathsf{z}(X\omega))\frac{\alpha}{\mathsf{Z_H}(X)} \\ &+ (\mathsf{z}(X) - 1)\mathsf{L}_1(X)\frac{\alpha^2}{\mathsf{Z_H}(X)} \end{aligned}$$

Split $\mathsf{t}(X)$ into degree less then n polynomials $\mathsf{t_{lo}}(X), \mathsf{t_{mid}}(X), \mathsf{t_{hi}}(X)$, such that

$$\mathsf{t}(X) = \mathsf{t_{lo}}(X) + X^{\mathsf{n}}\mathsf{t_{mid}}(X) + X^{2\mathsf{n}}\mathsf{t_{hi}}(X)\,.$$

Output $[\mathsf{t_{lo}}(\chi), \mathsf{t_{mid}}(\chi), \mathsf{t_{hi}}(\chi)]_1$.

Round 4 Get the challenge $\mathfrak{z} \in \mathbb{F}_p$, $\mathfrak{z} = \mathcal{H}(\pi[0..3])$. Compute opening evaluations

$$\mathsf{a}(\mathfrak{z}), \mathsf{b}(\mathfrak{z}), \mathsf{c}(\mathfrak{z}), \mathsf{S}_{\sigma 1}(\mathfrak{z}), \mathsf{S}_{\sigma 2}(\mathfrak{z}), \mathsf{t}(\mathfrak{z}), \mathsf{z}(\mathfrak{z}\omega),$$

Compute the linearisation polynomial

$$\begin{aligned} \mathsf{r}(X) = \; &\mathsf{a}(\mathfrak{z})\mathsf{b}(\mathfrak{z})\mathsf{q_M}(X) + \mathsf{a}(\mathfrak{z})\mathsf{q_L}(X) + \mathsf{b}(\mathfrak{z})\mathsf{q_R}(X) + \mathsf{c}(\mathfrak{z})\mathsf{q_O}(X) + \mathsf{q_C}(X) \\ &+ \alpha \cdot ((\mathsf{a}(\mathfrak{z}) + \beta\mathfrak{z} + \gamma)(\mathsf{b}(\mathfrak{z}) + \beta k_1 \mathfrak{z} + \gamma)(\mathsf{c}(\mathfrak{z}) + \beta k_2 \mathfrak{z} + \gamma) \cdot \mathsf{z}(X)) \\ &- \alpha \cdot ((\mathsf{a}(\mathfrak{z}) + \beta \mathsf{S}_{\sigma 1}(\mathfrak{z}) + \gamma)(\mathsf{b}(\mathfrak{z}) + \beta \mathsf{S}_{\sigma 2}(\mathfrak{z}) + \gamma)\beta \mathsf{z}(\mathfrak{z}\omega) \cdot \mathsf{S}_{\sigma 3}(X)) \\ &+ \alpha^2 \cdot \mathsf{L}_1(\mathfrak{z}) \cdot \mathsf{z}(X) \end{aligned}$$

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

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

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

$$\mathsf{W}_{\mathfrak{z}\omega}(X) = \frac{\mathsf{z}(X) - \mathsf{z}(\mathfrak{z}\omega)}{X - \mathfrak{z}\omega}$$

Output $\left[\mathsf{W}_{\mathfrak{z}}(\chi), \mathsf{W}_{\mathfrak{z}\omega}(\chi)\right]_1$.

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

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

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

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

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

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

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

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

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

Step 12 Check whether the verification equation holds

$$\left(\left[\mathsf{W}_{\mathfrak{z}}(\chi)\right]_1 + u \cdot \left[\mathsf{W}_{\mathfrak{z}\omega}(\chi)\right]_1\right) \bullet [\chi]_2 - \left(\mathfrak{z} \cdot \left[\mathsf{W}_{\mathfrak{z}}(\chi)\right]_1 + u\mathfrak{z}\omega \cdot \left[\mathsf{W}_{\mathfrak{z}\omega}(\chi)\right]_1 + [F]_1 - [E]_1\right) \bullet [1]_2 = 0\,. \tag{25}$$

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

Plonk **simulator** $\mathsf{Sim}_\chi(\mathsf{srs}, \mathsf{td} = \chi, \mathsf{x})$**:**
The Plonk simulator proceeds as an honest prover would, except:

1. In the first round, it sets $\mathsf{w} = (\mathsf{w}_i)_{i \in [1\,..\,3\mathsf{n}]} = \mathbf{0}$, and at random picks $b_1, \ldots, b_9$. Then it proceeds with that all-zero witness.
2. In Round 3, it computes polynomial $\mathsf{t}(X)$ honestly, however uses trapdoor χ to compute commitments $\mathsf{t}_{\mathsf{lo}}(\chi), \mathsf{t}_{\mathsf{mid}}(\chi), \mathsf{t}_{\mathsf{hi}}(\chi)$.