On Simulation-Extractability of Universal zkSNARKs

Anonymous submission to PKC

No Institute Given

Abstract. In this paper we show that a wide class of (computationally) special-sound proofs of knowledge which have unique response property and are standard-model zero-knowledge are (non-black-box) simulation-extractable when made non-interactive by the Fiat–Shamir transform. We prove that two efficient updatable universal zkSNARKs—Plonk [?] and Sonic [?]—meet these requirements and conclude by showing their simulation-extractability. As a side result we also show that relying security on rewinding and Fiat–Shamir transform often comes at a great price of inefficient (yet still polynomial time) knowledge extraction and the security loss introduced by these techniques should always be taken into account.

1 Introduction

1.1 Motivation

The rise of updatable zkSNARKs. In recent years we have seen indisputable progress in building efficient zero-knowledge proof systems, e.g. [?, ?, ?, ?, ?, ?, ?, ?] to name a few. Special attention has been devoted to the design of new *zero-knowledge succinct non-interactive arguments of knowledge* (zkSNARKs) with short (usually constant-length) proofs. This property makes zkSNARKs especially useful in real-life system deployment, e.g. [?, ?, ?, ?, ?]. This is turn risen a question—*Are zkSNARKs' security models useful in real life?* Answering that becomes especially important when one realises that all zkSNARKs with constant-sized proofs are shown secure in the common reference string (CRS) model where the CRS used by the parties—i.e. a prover that shows veracity of a statement using its private input called *witness* and a verifier that verifies it—comes with a trapdoor which can be used to break (knowledge) soundness; that is, convince the verifier to accept a false statement or a statement for which the prover does not know the witness. Hence for secure real-life deployment of zkSNARKs requires answering to the number of questions—*How can the parties be sure that the trapdoor has not leaked? Who is the party that generates the CRS? Can this party be trusted?* The last question has to be answered positively, as otherwise the CRS-generating party may leak the trapdoor. The questions are especially important in zero knowledge proofs used in distributed systems, like e.g. blockchains, where assuming that a trusted party generates the CRS may be a step too far and subvert the whole purpose of a decentralised system.

One of the first to tackled this question were Bellare et al. [?] who proposed notions of *subversion zero knowledge* and *subversion soundness.* The former states that zero knowledge property persists even when the CRS is produced by a malicious party. The latter states that the proof system remains sound in the same situation. Importantly, Bellare et al. shown that no system can be simultaneously subversion zero-knowledge and subversion-sound. Abdolmaleki et al. [?], and independently Fuchsbauer [?], shown that the most efficient zkSNARK for QAP[1] by Groth [?] can be made subversion zero-knowledge by making minor modifications to its CRS and without sacrificing its efficiency.

Although efficient, Groth's zkSNARK comes with a drawback—the CRS is relation-dependent. That is, if one wants to show that two different arithmetic circuits have been evaluated correctly, one has to do that using two different CRS-s. Since CRS generation is a troublesome process, it is desired to have it performed once, not every time a new circuit validity has to be shown. zkSNARKs that utilise a single CRS for all circuits of a given size are called *universal.* One of the first universal zkSNARK was proposed by Groth et al. [?]. This particular proof system also introduced a novel security property called *updatable soundness.* Because of the Bellare et al. [?] impossibility result, one cannot wish for a system that is simultaneously subversion zero-knowledge and subversion-sound. Groth et al. work around this problem by proposing a notion which allows zkSNARK provers and verifiers to *update* the CRS, i.e. to take a CRS and modify it in a well-defined and verifiable way to obtain a new CRS. Updates guarantee that if at least one of the CRS-updating parties is honest then the proof system is (knowledge) sound. Although inefficient, as the CRS length is quadratic to the size of the proven statements, [?] set a new paradigm for designing zkSNARKs.

The first universal zkSNARK with updatable and short CRS was Sonic proposed by Maller et al. in [?]. Eventually, 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. All these protocols utilise strong cryptographic assumptions like the algebraic group model (AGM) and the random oracle model (ROM) to show their security. They also use their own representation of circuits instead of the "standard" quadratic arithmetic programs (QAP). However, these protocols are also one of the most interesting schemes for practitioners due to their practicality stemming from the efficient proving and verification algorithms, and the universality of the CRS, as well as for their security model that diminishes the need for a trusted party to provide a CRS.

On the importance of the simulation extractability. Although zkSNARKs are shown to satisfy a (standard) knowledge soundness definition, simulation-extractability (SE) is the property that should be required from zkSNARKs used in practice. This is since, arguably, in the real life one simply cannot assume that

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

the adversary who tries to break security of a system does not have access to any proofs provided by other parties using the same zero-knowledge scheme. On the contrary, in the most popular applications of zkSNARKs, like privacy-preserving blockchains, proofs made by all blockchain-participants are usually public. Thus, it is only reasonable to require a zero-knowledge proof system to be resilient to attacks that utilise proofs generated by different parties.

There are many results on simulation extractability for non-interactive zero-knowledge proofs (NIZKs). First, Groth [?] noticed that a (black-box) simulation extractable 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 it is important to mention such works as the first simulation-extractable zkSNARK 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 as the statement's witness is encrypted. Recently, Abdolmaleki et al. [?] showed another transformation that obtains non-black-box simulation extractability but also preserves succinctness of the argument.

Independently, some authors focused on obtaining simulation extractability of known zkSNARKs, like GROTH16 [?], by introducing minor modifications and using stronger assumptions [?, ?]. Interestingly, although such modifications hurt performance of the proof system, the resulting zkSNARKs are still more efficient than the first SE zkSNARK [?], see [?]. Recently, [?] showed that the original Groth's proof system from [?] is weakly SE and randomisable.

State of the art—simulation-extractable updatable universal zkSNARKs. Up to our best knowledge, there are zkSNARKs that are simulation-extractable and zkSNARKs that are universal, however there are no known zkSNARKs that enjoy both of these properties out-of-the-box. Obviously, given a universal zkSNARK one could lift it to be simulation-extractable using techniques described e.g. in [?, ?], but such lift comes with inevitably efficiency loss. The same applies for updatable zkSNARKs. No out-of-the-box simulation-extractable updatable zkSNARKs are currently known and there is no transformation that could take a SE zkSNARK and make it updatable (even though transformations like [?] preserves updatability).

1.2 Our contribution

First of all, we show that a class of computationally special-sound proofs of knowledge that are zero-knowledge in the standard model and have a unique response property *are simulation-extractable out-of-the box* when the Fiat–Shamir transformation is applied to them. Although a similar problem has been already tackled by Faust et al. [?], we extend it to a much wider class of protocols, e.g. our result is not restricted to 3-message protocols only.

To show that our result is useful and practical we prove that two of the most efficient updatable and universal zkSNARKs—Plonk and Sonic—are simulation-extractable. Although—or arguably because—we do not change these protocols at all, we had to solve a number of problems in order to obtain this result which are explained below.

Before we continue, we note that Plonk and Sonic—as originally presented in [?] and [?]—are interactive proofs of knowledge made non-interactive by the Fiat–Shamir transform. In the following, we denote the underlying interactive protocols by $\mathbf{P}$ (for Plonk) and $\mathbf{S}$ (for Sonic) and the resulting, non-interactive ones, by $\mathbf{P}_{\mathsf{FS}}$ and $\mathbf{S}_{\mathsf{FS}}$, respectively.

Special soundness. First, following [?], we need to show that $\mathbf{P}$ and $\mathbf{S}$ are special-sound. However the standard definition of special soundness could not be met. First of all, the definition requires extraction of a witness from any two transcripts, each containing three messages and sharing the first message. For $\mathbf{P}$ and $\mathbf{S}$ that gives nothing. The definition had to be tuned to cover protocols that have more messages than just three. Furthermore, the number of transcripts required is much greater—$(\mathsf{n} + 3)$—where n is the number of constraints in the proven circuit—for $\mathbf{P}$ and $(\mathsf{n}{+}1)$–where n and Q are the numbers of multiplicative and linear constraints— for $\mathbf{S}$. Hence, we do not have a *pair of transcripts*, but a *tree of transcripts.*

Secondly, both protocols rely on common reference strings which come with trapdoors that allow an adversary who knows them to produce multiple valid proofs even without knowing the witness or for false statements. Recall that the standard special soundness definition requires witness extraction from *any* pair of acceptable transcripts that share a common root. Fortunately, one could define a weaker version of special soundness, i.e. a computational special soundness. That is, we show that either it is possible to extract a witness from a tree of acceptable transcripts or one could use the adversary that produced the tree to break some underlying computational assumption.

Unique response property. Another property that has to be proven is the unique response property which states, as expressed in [?], that except for the first message, all messages sent by the prover are deterministic (intuitively, the prover does not use its randomness except from the first message). As previously, we also could not use this definition right out of the box simply because $\mathbf{P}$ does not follow it—both first and the second prover's messages are randomised. We thus propose a generalisation of the definition which states that a protocol is i-ur if the prover is deterministic after sending its i-th message. This property is fulfilled by $\mathbf{P}$ with $i = 2$.

To be able to show the unique response property (for both of the protocols) we also had to show that the modified KZG polynomial commitment schemes [?] proposed in [?] and [?] have a *unique opening property* which states that for a polynomial $\mathsf{f}(X)$ evaluated at some point z it should be infeasible for any PPT adversary to provide two different (even honest) but acceptable openings of the commitment.

HVZK. In order to show our result we also show that (interactive) P and S are honest verifier zero-knowledge in the standard model, i.e. the simulator is able to produce a transcript indistinguishable from a transcript produced by an honest prover and verifier without any additional knowledge, esp. without knowing the CRS trapdoor. Although both Sonic and Plonk are shown to be zero-knowledge, the proofs provided by their authors utilise trapdoors. For our reduction to work, we need simulators that provide indistinguishable proofs relying only on rearranging the ordering of messages and picking suitable verifier's challenges. That is, any PPT party should be able to produce a simulated proof by its own. (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). This property allows us to build simulators for P_{FS} and S_{FS} that rely only on programmability of the random oracle.

Generalisation of the general forking lemma. Consider an interactive 3-message special-sound protocol $\mathbf{\Psi}$ and its non-interactive version $\mathbf{\Psi}_{\mathsf{FS}}$ obtained by the Fiat–Shamir transform. The general forking lemma provides an instrumental lower bound for probability of extracting a witness from 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, as known from [?], cannot be used directly. We thus propose a modification 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. (That said, we have to note that the security loss is polynomial, albeit big.)

We note that some modern zkSNARKs, like [?, ?], rely on the Fiat–Shamir transform and the forking lemma heavily. First, an interactive protocol is proposed and its security and special-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 generalisation of the forking lemma or disclose a transformation that does not suffer from the generalisation inefficiency. We note that the security loss may also apply when knowledge soundness is proven. That is the case for Sonic, which security proof relies on so-called witness-extended emulation. Authors of Plonk worked around this problem by showing their protocol secure directly in the algebraic group model.

Towards simulation-extractability. Given our modified, less restrictive, definition for special-soundness and the unique response property, and our generalised forking lemma we are able to show the announced result—simulation extractability of P_{FS} and S_{FS}. The proof is inspired by simulation-extractability and simulation-soundness proofs from [**?**], with major modifications, which were required as [**?**] considers (Fiat–Shamir transformed) Σ-protocols that are undoubtedly simpler protocols than the considered proof systems.

1.3 Structure of the paper

In the next section we present necessary preliminaries. Section 3 compounds of new notions and theorems that instantiate our framework. Then, in Section 4, we show our main result, that is a proof of simulation-soundness for a class of zero-knowledge proofs of knowledge. In Section 5 we show that Plonk fulfils requirements of our framework and in fact is simulation extractable. Due to the lack of space, the analogous result for Sonic has been moved to Supp. Mat. C.

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 $|a(\lambda) - b(\lambda)| \leq \mathsf{negl}(\lambda)$.

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 $a g_\iota$ 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.

Security of Plonk and Sonic relies on two discrete-log based security assumptions—(q_1, q_2)-dlog assumption and its extended with negative exponents version (q_1, q_2)-ldlog assumption[2]. Due to the lack of space, the assumptions have been moved to Supp. Mat. A.

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 the *uber assumption* introduced by Boneh et al. [?] as presented by Boyen in [?].

Let $r, s, t, c \in \mathbb{N}\backslash\{0\}$, Consider tuples 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 $\mathsf{T} = (\mathsf{t}_1, \ldots, \mathsf{t}_r)$ 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 1 (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.*

For the sake of this paper we modify the assumption slightly, i.e. we require not target group $\mathbb{G}_T$ elements to be indistinguishable, but elements of $\mathbb{G}_1$, more precisely we require:

Definition 2 ($(\mathsf{R}, \mathsf{S}, \mathsf{T}, \mathsf{f})$-uber assumption). *Let $\mathsf{R}, \mathsf{S}, \mathsf{T}$ be defined as above, $(x', x_1, \ldots, x_c) \leftarrow\!\!\$\ \mathbb{F}_p^{c+1}$ and let f be independent on $(\mathsf{R}, \mathsf{S}, \mathsf{T})$, cf. Definition 1. Then, for any PPT adversary $\mathcal{A}$*

$$\begin{aligned}\Pr[\mathcal{A}([\mathsf{R}(x_1, \ldots x_c)]_1, [\mathsf{S}(x_1, \ldots, x_c)]_2, [\mathsf{T}(x_1, \ldots, x_c)]_T, [\mathsf{f}(x_1, \ldots, x_c)]_1) = 1] \approx_\lambda \\ \Pr[\mathcal{A}([\mathsf{R}(x_1, \ldots x_c)]_1, [\mathsf{S}(x_1, \ldots, x_c)]_2, [\mathsf{T}(x_1, \ldots, x_c)]_T, [x']_1) = 1].\end{aligned}$$

For the original formulation of the BBG uber-assumption see Definition 9 in Supp. Mat. A.

Proofs by Game-Hoping. Proofs by *game hoping* 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 $\mathsf{A} \land \overline{\mathsf{F}} \iff \mathsf{B} \land \overline{\mathsf{F}}$. Then $|\Pr[\mathsf{A}] - \Pr[\mathsf{B}]| \le \Pr[\mathsf{F}]$.*

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

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

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

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

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

Fig. 1: $\mathbf{PC_P}$ polynomial commitment scheme

2.2 Polynomial commitment.

In the polynomial commitment scheme $\mathbf{PC} = (\mathsf{KGen}, \mathsf{Com}, \mathsf{Op}, \mathsf{Vf})$ the committer C can prove the receiver R that some polynomial f which C committed to evaluates to s at some point z chosen by R. Plonk and Sonic use variants of the KZG polynomial commitment scheme. We denote the first by $\mathbf{PC_P}$, presented in Fig. 1, and the latter by $\mathbf{PC_S}$, presented in Fig. 3. (Due to the lack of space, Fig. 3 was moved to Supp. Mat. A.) We emphasize the following properties of a secure polynomial commitment $\mathbf{PC}$:

Evaluation binding a PPT adversary $\mathcal{A}$ which outputs a commitment $\boldsymbol{c}$ and evaluation points $\boldsymbol{z}$ has at most negligible chances 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 $\mathcal{A}$

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

We say that $\mathbf{PC}$ has the unique opening property if the following holds:

Opening uniqueness Let $k \in \mathbb{N}$ be the number of committed polynomials, $l \in \mathbb{N}$ number of evaluation points, $\boldsymbol{c} \in \mathbb{G}^k$ be the commitments, $\boldsymbol{z} \in \mathbb{F}_p^l$ be the arguments the polynomials are evaluated at, $\boldsymbol{s} \in \mathbb{F}_p^k$ the evaluations, and $\boldsymbol{o} \in \mathbb{F}_p^l$ be the commitment openings. Then for every PPT adversary $\mathcal{A}$

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

Intuitively, opening uniqueness assures that there is only one valid opening for the committed polynomial and given evaluation point. This property is crucial in showing simulation-extractability of Plonk and Sonic. We show that the Plonk's and Sonic's polynomial commitment schemes satisfy this requirement in Lemma 3 and Lemma 8 respectively.

Commitment of knowledge For every PPT adversary $\mathcal{A}$ there exists a PPT extractor Ext such that

$$\Pr[\mathsf{f} = \mathsf{Ext}_{\mathcal{A}}(\mathsf{crs}, c), c = \mathsf{Com}(\mathsf{crs}, \mathsf{f}) \mid \mathsf{crs} \leftarrow \mathsf{KGen}, c \leftarrow \mathcal{A}(\mathsf{crs})] \geq 1 - \varepsilon_{\mathsf{k}}(\lambda)$$

In that case we say that $\mathbf{PC}$ is ε_{k}-knowledge.

Intuitively when a commitment scheme is a commitment 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 $\mathbf{PC}_{\mathbf{S}}$ is a commitment of knowledge. The same reasoning could be used to show that property for $\mathbf{PC}_{\mathbf{P}}$. We skip a proof for that fact due to the lack of space.

2.3 Algebraic Group Model

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

2.4 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 crs. We denote by $\langle \mathsf{P}(\mathbf{R}, \mathsf{crs}, \mathsf{x}, \mathsf{w}), \mathsf{V}(\mathbf{R}, \mathsf{crs}, \mathsf{x})\rangle$ a transcript trans (or, proof π—we use these two alternatively) of a conversation between a P with input $(\mathbf{R}, \mathsf{crs}, \mathsf{x}, \mathsf{w})$ and V with input $(\mathbf{R}, \mathsf{crs}, \mathsf{x})$. We write $\langle \mathsf{P}(\mathbf{R}, \mathsf{crs}, \mathsf{x}, \mathsf{w}), \mathsf{V}(\mathbf{R}, \mathsf{crs}, \mathsf{x})\rangle = 1$ if in the end of the transcript the verifier V returns 1 and say that V accepts the transcript. We sometimes abuse notation and write $\mathsf{V}(\mathsf{trans}) = 1$ to denote a fact that trans is accepted by the verifier.

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

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

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

Soundness We say that $\boldsymbol{\Psi}$ 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, i.e for $\mathsf{x} \notin \mathcal{L}$. More precisely, for all $\mathbf{R} \in \mathcal{R}(1^\lambda)$

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

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

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

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

A CRS crs may come with a secret string called *trapdoor* td that allows the simulator to simulate. In this paper we distinguish simulators that requires a trapdoor to be able to provide an indistinguishable proof and those that do not. We call the former *CRS-simulators* and denote by Sim_td for trapdoor td.

Occasionally, a weaker version of zero knowledge is required, so called *honest verifier zero knowledge* (HVZK) which assumes that the verifier's challenges are picked at random from some predefined set. Although weaker, this definition suffices in many applications. Especially, an interactive zero-knowledge proof that is HVZK and *public-coin* (i.e. the verifier outputs as challenges its random coins) can be made non-interactive in the random oracle model by using the Fiat–Shamir transformation.

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 formalised by the following notion:

Knowledge soundness We call an interactive proof system $\mathbf{\Psi}$ *knowledge-sound* if for any $\mathbf{R} \in \mathcal{R}(1^\lambda)$ and a PPT adversary $\mathcal{A}$

$$\Pr\left[\begin{array}{l} \mathsf{V}(\mathbf{R}, \mathsf{crs}, \mathsf{x}, \mathsf{trans}) = 1 \\ \wedge\, \mathbf{R}(\mathsf{x}, \mathsf{w}) = 0 \end{array} \middle| \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{KGen}(\mathbf{R}), \mathsf{x} \leftarrow \mathcal{A}(\mathbf{R}, \mathsf{crs}), \\ (\mathsf{w}, \mathsf{trans}) \leftarrow \mathsf{Ext}^{\langle \mathcal{A}(\mathbf{R}, \mathsf{crs}, \mathsf{x}), \mathsf{V}(\mathbf{R}, \mathsf{crs}, \mathsf{x}) \rangle}(\mathbf{R}, \mathsf{x}) \end{array}\right] \leq \mathsf{negl}(\lambda)\,,$$

NIZKs in the Random Oracle Model.

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 which transcript compounds of three messages (a, b, z), the middle being 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 $b \neq b'$. Formally,

Special soundness A sigma protocol Σ is *specially-sound* if for any adversary $\mathcal{A}$ the probability

$$\Pr\left[\begin{array}{l} \mathsf{w} \leftarrow \mathsf{Ext}(\mathbf{R}, \mathsf{x}, (a, b, z), (a, b', z')), \\ \mathbf{R}(\mathsf{x}, \mathsf{w}) = 0 \end{array} \middle| \begin{array}{l} (\mathsf{x}, (a, b, z), (a, b', z')) \leftarrow \mathcal{A}(\mathbf{R}), \\ \mathsf{V}(\mathbf{R}, \mathsf{x}, (a, b, z)) = \\ \qquad = \mathsf{V}(\mathbf{R}, \mathsf{x}, (a, b', z')) = 1, \end{array}\right] \leq \mathsf{negl}(\lambda)\,.$$

Another property that sigma protocols sometimes 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 $\mathbf{R} \in \mathcal{R}(1^\lambda)$ which proofs compound of three messages (a, b, z). We say that Σ is has a unique response property if for all PPT algorithms $\mathcal{A}$ holds

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

If this property holds even against unbounded adversaries, it is called *strict*, cf. [?]. Later on we often 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 requirement.

2.5 Simulation extractable NIZKs from sigma protocols

Real life applications often require from 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. A strong version of non-malleability is formalised by simulation extractability which assures that no adversary can produce 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 (Simulation-extractable NIZK). *Let* $\boldsymbol{\Psi} = (\mathsf{KGen}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *be a computationally special-sound HVZK proof and* $\boldsymbol{\Psi}_{\mathsf{FS}} = (\mathsf{KGen}_{\mathsf{FS}}, \mathsf{P}_{\mathsf{FS}}, \mathsf{V}_{\mathsf{FS}}, \mathsf{Sim}_{\mathsf{FS}})$ *be* $\boldsymbol{\Psi}$ *transformed by the Fiat–Shamir transform. We say that* $\boldsymbol{\Psi}_{\mathsf{FS}}$ *is* 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* $\boldsymbol{\Psi}$ *with probability* acc*, that is*

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

probability

$$\mathsf{frk} = \Pr\left[\begin{array}{l|l} \mathsf{V}_{\mathsf{FS}}(\mathbf{R}, \mathsf{crs}, \mathsf{x}_{\mathcal{A}_{\mathsf{se}}}, \pi_{\mathcal{A}_{\mathsf{se}}}) = 1, & \mathsf{crs} \leftarrow \mathsf{KGen}(\mathbf{R}), r \leftarrow\!\!\$\ \mathsf{R}(\mathcal{A}_{\mathsf{se}}), \\ (\mathsf{x}_{\mathcal{A}_{\mathsf{se}}}, \pi_{\mathcal{A}_{\mathsf{se}}}) \notin Q, & (\mathsf{x}_{\mathcal{A}_{\mathsf{se}}}, \pi_{\mathcal{A}_{\mathsf{se}}}) \leftarrow {\mathcal{A}_{\mathsf{se}}}^{\mathsf{Sim}_{\mathsf{FS}}, \mathcal{H}}(\mathbf{R}, \mathsf{crs}; r) \\ \mathbf{R}(\mathsf{x}_{\mathcal{A}_{\mathsf{se}}}, \mathsf{w}_{\mathcal{A}_{\mathsf{se}}}) = 0 & \mathsf{w}_{\mathcal{A}_{\mathsf{se}}} \leftarrow \mathsf{Ext}_{\mathsf{ss}}(\mathbf{R}, \mathsf{crs}, \mathcal{A}_{\mathsf{se}}, r, \mathsf{x}_{\mathcal{A}_{\mathsf{se}}}, \pi_{\mathcal{A}_{\mathsf{se}}}, Q, Q_{\mathcal{H}},) \end{array}\right]$$

is at at least

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

for some polynomial $\mathsf{poly}(\lambda)$*, constant d and negligible ε 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* $\mathcal{A}_{\mathsf{se}}$*'s queries to* $\mathcal{H}$ *and* $\mathcal{H}$*'s answers.*

Consider a sigma protocol $\boldsymbol{\Sigma} = (\mathsf{P}, \mathsf{V}, \mathsf{Sim})$ that is specially sound and has a unique response property. Let $\boldsymbol{\Sigma}_{\mathsf{FS}} = (\mathsf{P}_{\mathsf{FS}}, \mathsf{V}_{\mathsf{FS}}, \mathsf{Sim}_{\mathsf{FS}})$ be a NIZK obtained by applying the Fiat–Shamir transform to $\boldsymbol{\Sigma}$. Faust et al. [**?**] show that every such $\boldsymbol{\Sigma}_{\mathsf{FS}}$ is simulation-extractable. This result is presented in Supp. Mat. B along with the instrumental forking lemma, cf. [**?**].

3 Towards simulation extractability for multi-message protocols, definitions and lemmas

Unfortunately, Faust et al.'s result cannot be directly applied in our case since the protocols we consider have more than three messages, require more than just two transcript for the extractor to work and are not special sound.

3.1 Generalised forking lemma.

First of all, although dubbed "general", Lemma 7 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 $\mathsf{n}+3$ valid proofs. Here we propose a generalisation of the general forking lemma that given probability of producing an accepting transcript $\widetilde{\mathsf{acc}}$ 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 $\mathbf{\Psi}$. A $(n_1, \ldots, n_\mu)$-tree of accepting transcript is a tree where each node on depth i, for $i \in [1 \mathinner{..} \mu]$, is an i-th prover's message in an acceptable transcripts; edges between the nodes are labeled with verifier's challenges; and each node on depth i has $(n_i - 1)$ siblings and n_{i+1} children. Altogether, the tree consists of $N = \prod_{i=1}^{\mu} n_i$ branches, which makes N acceptable transcripts. We require $N = \mathsf{poly}(\lambda)$.*

We note that the general forking lemma proposed in Lemma 2 works for protocols which have extractable witness from a $(1, \ldots, 1, n_i, 1, \ldots, 1)$-tree of acceptable transcripts. This limitation however does not affect the main result of this paper, i.e. showing that both Plonk and Sonic are simulation extractable.

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 $i \in [0 \mathinner{..} q]$ and s is called a side output. Denote by IG a randomised instance generator. We denote by acc the probability*

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

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

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

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

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

We start by claiming that for all y

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

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

$$\mathsf{frk} = \mathbb{E}\left[\mathsf{frk}(y)\right] \geq \mathbb{E}\left[\frac{\mathsf{acc}(y)^m}{q^{m-1}} - \mathsf{acc}(y) \cdot \left(1 - \frac{h!}{(h-m)! \cdot h^m}\right)\right] \tag{2}$$

```
$\mathsf{GF}^m_{\mathcal{Z}}(y, h^1_1, \ldots, h^1_q)$
$\rho \leftarrow\!\!\$\ \mathsf{R}(\mathcal{Z})$
$(i, s_1) \leftarrow \mathcal{Z}(y, h^1_1, \ldots, h^1_q; \rho)$
**if** $i = 0$ **return** $(0, \perp)$
**for** $j \in [2 \mathinner{..} m]$
  $h^j_1, \ldots, h^j_{i-1} \leftarrow h^{j-1}_1, \ldots, h^{j-1}_{i-1}$
  $h^j_i, \ldots, h^j_q \leftarrow\!\!\$\ H$
  $(i', s_j) \leftarrow \mathcal{Z}(y, h^j_1, \ldots, h^j_{i-1}, h^j_i, \ldots, h^j_q; \rho)$
  **if** $i' = 0 \vee i' \neq i$ **return** $(0, \perp)$
**if** $\exists (k, j), (k', j') \in [1 \mathinner{..} q] \times [1 \mathinner{..} m] \wedge (h^j_k = h^{j'}_{k'})$ **return** $(0, \perp)$
**else return** $(1, \boldsymbol{s})$
```

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

$$\geq \frac{\mathbb{E}\left[\mathsf{acc}(y)\right]^m}{q^{m-1}} - \mathbb{E}\left[\mathsf{acc}(y)\right] \cdot \left(1 - \frac{h!}{(h-m)! \cdot h^m}\right) \tag{3}$$

$$= \frac{\mathsf{acc}^m}{q^{m-1}} - \mathsf{acc} \cdot \left(1 - \frac{h!}{(h-m)! \cdot h^m}\right) . \tag{4}$$

Where Eq. (2) comes from Eq. (1); Eq. (3) comes from linearity of expected value and Lemma 12; and Eq. (4) holds by the fact that $\mathbb{E}\left[\mathsf{acc}(y)\right] = \mathsf{acc}$.

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

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

It remains to show that $\Pr[i_j = i_{j'} \wedge i_j \geq 1 \text{ for } (j,j') \in J] \geq \mathsf{acc}(y)^m / q^{m-1}$. Let $\mathsf{R}(\mathcal{Z})$ denote the set from which $\mathcal{Z}$ picks its coins at random. For each $\iota \in [1 \mathinner{..} q]$ 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

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

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

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

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

By Lemma 12 we get

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

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 from Lemma 13 where we set $x_i = \mathbb{E}[X_i]$, $y_i = 1$, $p = m$, and $q = m/(m-1)$ obtaining

$$\left(\sum_{i=1}^{q} \mathbb{E}[X_i]\right)^m \leq \left(\sum_{i=1}^{q} \mathbb{E}[X_i]^m\right) \cdot q^{m-1} \tag{5}$$

$$\frac{1}{q^{m-1}} \cdot \mathsf{acc}(y)^m \leq \sum_{i=1}^{q} \mathbb{E}[X_i]^m \; . \tag{6}$$

Finally, we get

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

□

We note that in the original forking lemma the forking algorithm F cf. Fig. 4 gets only as input y and elements $h_1^1, \ldots, h_q^1$ are randomly picked from H internally by F. However, assuming that $h_1^1, \ldots, h_q^1$ are random oracle responses, thus are random, makes our change only notational.

3.2 Unique-response protocols.

Another problem comes with another assumption required by Faust et al. That is, the unique response property of the transformed sigma protocol. Fischlin's formulation, although perfectly fine for applications presented in [?, ?], is not enough in our case. First of all, the property assumes that the protocol has three messages, with the middle 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 could not work since if there is more challenges the prover's messages has to respond to them. Another problem rises when the protocol contains some message—obviously, except the first one—where the prover randomises its message. In that case unique-responsiveness can not hold as well. Last but not least, the protocols we consider here are not designed to be in the standard model, but utilises CRS. That 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 after* i, and called it an i-ur-protocol, if it follows the definition below:

Definition 5 (i-ur-protocol). *Let* $\boldsymbol{\Psi}$ *be a* $(2\mu + 1)$*-message proof system* $\boldsymbol{\Psi} = (\mathsf{KGen}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$*. Let* $\boldsymbol{\Psi}_{\mathsf{FS}}$ *be* $\boldsymbol{\Psi}$ *after Fiat–Shamir transform, Denote by* $a_0, \ldots, a_\mu$ *protocol messages output by the prover, We say that* $\boldsymbol{\Psi}$ *has* unique responses after i *if for any* PPT *adversary* $\mathcal{A}$*:*

$$\Pr\left[\begin{array}{l|c} \boldsymbol{a} = (a_0, \ldots, a_\mu), \boldsymbol{a'} = (a'_0, \ldots, a'_\mu) \leftarrow \mathcal{A}^{\mathcal{H}}(\mathbf{R}, \mathsf{crs}), \boldsymbol{a} \neq \boldsymbol{a'} & \\ \mathsf{V}^{\mathcal{H}}_{\mathsf{FS}}(\mathbf{R}, \mathsf{crs}, \mathsf{x}, \boldsymbol{a}) = \mathsf{V}^{\mathcal{H}}_{\mathsf{FS}}(\mathbf{R}, \mathsf{crs}, \mathsf{x}, \boldsymbol{a'}) = 1, & \mathsf{crs} \leftarrow \mathsf{KGen}_{\mathsf{FS}}(\mathbf{R}) \\ a_j = a'_j,\ j \in [1 \mathinner{..} i+1] & \end{array}\right]$$

is at most negligible at λ*.*

Intuitively, a protocol is i-ur if after i-th prover's message, all P's further messages are determined by the witness it knows, the messages it already send and received and the future challenges from the verifier. We note that the definition above is independent on whether the proof system $\boldsymbol{\Psi}$ utilises CRS (and compounds of the CRS-generating KGen algorithm) or not.

3.3 rewinding-based knowledge soundness

Note that the special soundness property (as usually defined) holds for all—even computationally unbounded—adversaries. Since a simulation trapdoor for $\mathsf{\textbf{P}}$ exists, it is not special sound in that regard—as an unbounded adversary could reveal the trapdoor and build a number of simulated proofs for a fake statement. Hence, we provide a weaker, yet sufficient, definition of *rewinding-based knowledge 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.

Definition 6 (rewinding-based knowledge soundness). *Let* $\boldsymbol{\Psi} = (\mathsf{KGen}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *be an* $(2\mu + 1)$*-message proof system for a relation* $\mathbf{R}$*. We say that* $\boldsymbol{\Psi}$ *is* $(n_1, \ldots, n_\mu)$-special sound *if for every* PPT *adversary* $\mathcal{A}(\mathbf{R}, \mathsf{crs})$*,* *where* $\mathsf{crs} \leftarrow \mathsf{KGen}(\mathbf{R})$*, that produces an accepting* $(n_1, \ldots, n_\mu)$*-tree of transcripts* T *for a statement* x *there exists an extractor* Ext *that given* T *extracts* w *such that* $\mathbf{R}(\mathsf{x}, \mathsf{w}) = 1$ *with an overwhelming probability.*

Since we do not utilise the classical special soundness (that holds for all, even unbounded, adversaries) all references to that property should be understood as references to its computational version.

4 Simulation-extractability—the general result

Theorem 1 (Simulation-extractable multi-message protocols). *Let* $\mathbf{\Psi} = (\mathsf{KGen}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *be an interactive* $(2\mu + 1)$*-message proof system that is honest verifier zero-knowledge in the standard model*[3]*, has* k*-*ur *property with security* $\varepsilon_{\mathsf{ur}}$*, is* $(1, \ldots, 1, n_j, 1, \ldots, 1)$*-special sound, for some* $j > k$*.*

Let $\mathcal{H}\colon \{0,1\}^* \to \{0,1\}^\lambda$ *be a random oracle. Then* $\mathbf{\Psi}_{\mathsf{FS}}$ *is simulation-extractable with extraction error* $\varepsilon_{\mathsf{ur}}$ *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*

$$\mathsf{ext} \geq \frac{1}{q^{n_j - 1}} (\mathsf{acc} - \varepsilon_{\mathsf{ur}})^{n_j} - \varepsilon \,,$$

for some negligible ε*.*

Proof. The proof goes by game hoping. The games are controlled by an environment $\mathcal{E}$ that internally runs a simulation extractability adversary $\mathcal{A}_{\mathsf{se}}$, 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 $\pi_{\mathcal{A}_{\mathsf{se}}}, \pi_{\mathsf{Sim}}$ 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, 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, we assume that 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.

Game G_0**:** This is a simulation extraction game played between an adversary $\mathcal{A}_{\mathsf{se}}$ who has given access to a random oracle $\mathcal{H}$ and simulator $\mathbf{\Psi}_{\mathsf{FS}}.\mathsf{Sim}$. There

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

is also an extractor Ext that, from the proof $\pi_{\mathcal{A}_{\mathsf{se}}}$ for instance $\mathsf{x}_{\mathcal{A}_{\mathsf{se}}}$ output by the adversary and from a transcripts of $\mathcal{A}_{\mathsf{se}}$'s operations, is tasked to extract a witness $\mathsf{w}_{\mathcal{A}_{\mathsf{se}}}$ such that $\mathbf{R}(\mathsf{x}_{\mathcal{A}_{\mathsf{se}}}, \mathsf{w}_{\mathcal{A}_{\mathsf{se}}})$ holds. $\mathcal{A}_{\mathsf{se}}$ 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 $\mathsf{x}_{\mathcal{A}_{\mathsf{se}}}$ such that $(\mathsf{x}_{\mathcal{A}_{\mathsf{se}}}, \pi_{\mathsf{Sim}}[1\,..\,k+1]) = (\mathsf{x}_{\mathcal{A}_{\mathsf{se}}}, \pi_{\mathcal{A}_{\mathsf{se}}}[1\,..\,k+1])$. That is, the adversary in its final proof reuses a part of a simulated proof it saw before and the proof is acceptable. Denote that event by $\mathsf{Err}_{\mathsf{ur}}$.

$\mathsf{G}_0 \mapsto \mathsf{G}_1$**:** We have, $\Pr\left[\mathsf{G}_0 \land \overline{\mathsf{Err}_{\mathsf{ur}}}\right] = \Pr\left[\mathsf{G}_1 \land \overline{\mathsf{Err}_{\mathsf{ur}}}\right]$ 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 $\mathcal{A}_{\mathsf{se}}$ winning it should be shown that $\Pr[\mathsf{Err}_{\mathsf{ur}}]$ is small.

Assume that $\mathcal{A}_{\mathsf{se}}$ queried the simulator on the instance it wishes to outputs $\mathsf{x}_{\mathcal{A}_{\mathsf{se}}}$. We show a reduction $\mathcal{R}_{\mathsf{ur}}$ that utilises $\mathcal{A}_{\mathsf{se}}$, who outputs a valid proof for $\mathsf{x}_{\mathcal{A}_{\mathsf{se}}}$, to break the k-ur property of $\mathbf{\Psi}$. Let $\mathcal{R}_{\mathsf{ur}}$ run $\mathcal{A}_{\mathsf{se}}$ internally as a black-box:

- The reduction answers both queries to the simulator $\mathbf{\Psi}_{\mathsf{FS}}.\mathsf{Sim}$ and to the random oracle. It also keeps lists Q, for the simulated proofs, and $Q_{\mathcal{H}}$ for the random oracle queries.
- When $\mathcal{A}_{\mathsf{se}}$ outputs a fake proof $\pi_{\mathcal{A}_{\mathsf{se}}}$ for $\mathsf{x}_{\mathcal{A}_{\mathsf{se}}}$, $\mathcal{R}_{\mathsf{ur}}$ looks through lists Q and $Q_{\mathcal{H}}$ until it finds $\pi_{\mathsf{Sim}}[0\,..\,k+1]$ such that $\pi_{\mathcal{A}_{\mathsf{se}}}[0\,..\,k+1] = \pi_{\mathsf{Sim}}[0\,..\,k+1]$ and a random oracle query $\pi_{\mathsf{Sim}}[k+1].\mathsf{ch}$ on $\pi_{\mathsf{Sim}}[0\,..\,k+1]$.
- $\mathcal{R}_{\mathsf{ur}}$ returns two proofs for $\mathsf{x}_{\mathcal{A}_{\mathsf{se}}}$:

$$\pi_1 = (\pi_{\mathsf{Sim}}[1\,..\,k+1], \pi_{\mathsf{Sim}}[k+1].\mathsf{ch}, \pi_{\mathsf{Sim}}[k+2\,..\,\mu])$$
$$\pi_2 = (\pi_{\mathsf{Sim}}[1\,..\,k+1], \pi_{\mathsf{Sim}}[k+1].\mathsf{ch}, \pi_{\mathcal{A}_{\mathsf{se}}}[k+2\,..\,\mu])$$

If $\pi_1 = \pi_2$, then $\mathcal{A}_{\mathsf{se}}$ fails to break simulation extractability, 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 $\mathbf{\Psi}$, what may happen with some negligible probability $\varepsilon_{\mathsf{ur}}$ only, hence $\Pr[\mathsf{Err}_{\mathsf{ur}}] \leq \varepsilon_{\mathsf{ur}}$.

Game G_2: This is identical to G_1 except that now the environment aborts also when it fails to build a $(1, \ldots, 1, n_j, 1, \ldots, 1)$-tree of accepting transcripts T by rewinding $\mathcal{A}_{\mathsf{se}}$. Denote that event by $\mathsf{Err}_{\mathsf{frk}}$. Note that for every acceptable proof $\pi_{\mathcal{A}_{\mathsf{se}}}$, we may assume that whenever $\mathcal{A}_{\mathsf{se}}$ outputs in Round $k+1$, a message $\pi_{\mathcal{A}_{\mathsf{se}}}[k+1]$, then $(\mathsf{x}_{\mathcal{A}_{\mathsf{se}}}, \pi_{\mathcal{A}_{\mathsf{se}}}[1\,..\,k+1])$ random oracle query that was made by the adversary, not the simulator[4], i.e. there is no simulated proof π_{Sim} on $\mathsf{x}_{\mathsf{Sim}}$ such that $(\mathsf{x}_{\mathcal{A}_{\mathsf{se}}}, \pi_{\mathcal{A}_{\mathsf{se}}}[1\,..\,k+1]) = (\mathsf{x}_{\mathsf{Sim}}, \pi_{\mathsf{Sim}}[1\,..\,k+1])$. Otherwise, the game would be already interrupted by the error event in Game G_1.

$\mathsf{G}_1 \mapsto \mathsf{G}_2$**:** As previously, $|\Pr[\mathsf{G}_1] - \Pr[\mathsf{G}_2]| \leq \Pr[\mathsf{Err}_{\mathsf{frk}}]$. Denote by $\widetilde{\mathsf{acc}}$ the probability that $\mathcal{A}_{\mathsf{se}}$ outputs a proof that is accepted and does not break k-ur-ness of $\mathbf{\Psi}$. Assuming that $\mathcal{A}_{\mathsf{se}}$ does not break the unique response property is necessary

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

to be able to use the forking lemma in which $\widetilde{\mathsf{acc}}'$ corresponds to the probability of an algorithm $\mathcal{Z}$ producing an accepting proof with a fresh challenge in round j. Otherwise, if k-ur-ness for $k < j$ does not hold and the adversary could, say, rerandomise a proof π it obtained from the simulator then it could force $\mathcal{Z}$ to never accept. The unique response property takes care of that. If an adversary would like to reuse a simulated proof it saw, it would end up with exactly the same proof which can also can not break simulation extractability, thus $\widetilde{\mathsf{acc}} = \widetilde{\mathsf{acc}}'$.

To explain importance of the unique response property we describe our extractor here. The extractor takes as input relation $\mathbf{R}$, CRS crs, $\mathcal{A}_{\mathsf{se}}$'s code, its randomness r, the output instance $\mathsf{x}_{\mathcal{A}_{\mathsf{se}}}$ and proof $\pi_{\mathcal{A}_{\mathsf{se}}}$, 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 $\mathsf{GF}_{\mathcal{Z}}^{n_j+1}(y, h_1, \ldots, h_q)$ for $y = (\mathcal{A}_{\mathsf{se}}, r, \mathsf{x}_{\mathcal{A}_{\mathsf{se}}}, \pi_{\mathcal{A}_{\mathsf{se}}}, Q)$ where we set $h_1, \ldots, h_q$ to be the consecutive queries from list $Q_{\mathcal{H}}$. We run $\mathcal{A}_{\mathsf{se}}$ internally in $\mathcal{Z}$.

To assure that in the first execution of $\mathcal{Z}$ the adversary $\mathcal{A}_{\mathsf{se}}$ produce the same $\mathsf{x}_{\mathcal{A}_{\mathsf{se}}}, \pi_{\mathcal{A}_{\mathsf{se}}}$ as in the extraction game, $\mathcal{Z}$ provides $\mathcal{A}_{\mathsf{se}}$ with the same randomness r and answers queries to the random oracle and simulator with responses prerecorded responses in $Q_{\mathcal{H}}$ and Q. Note, that since the view of the adversary run inside $\mathcal{Z}$ is the same as its view with access to real random oracle and simulator, it produces exactly the same output. After the first run, $\mathcal{Z}$ outputs the index of a random oracle query that was used by $\mathcal{A}_{\mathsf{se}}$ to compute the challenge $\mathcal{H}(\pi_{\mathcal{A}_{\mathsf{se}}}[0 \mathinner{..} j-1].\mathsf{ch})$ it had to answer in the j-th round and adversary's transcript, in GF description denoted by s_1. Then new random oracle responses are picked and the adversary is rewound to the point just prior the response to RO query $(\pi_{\mathcal{A}_{\mathsf{se}}}[0 \mathinner{..} j-1])$. 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 $\mathcal{H}(\pi_{\mathcal{A}_{\mathsf{se}}}[0 \mathinner{..} j-1])$, and a new transcript form the adversary s_2. $\mathcal{Z}$ is run with different random oracle responses till $n_j + 1$ valid transcripts are collected.

If the unique response property does not hold then in the j-th execution of $\mathcal{Z}$ the adversary could reuse a challenge that it learnt from observing proofs in Q. In that case, $\mathcal{Z}$ would not be able to output $i > 0$ and have to output 0, what would make extractor fail.

From the generalised forking lemma, cf. Lemma 2,

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

For the sake of simplicity we loose this approximation a bit and state

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

Game G_3**:** This is identical to G_2 except that now the environment uses the tree T to extract the witness for the proven statement and aborts when it fails. Denote that event by $\mathsf{Err}_{\mathsf{ss}}$.

$\mathsf{G}_2 \mapsto \mathsf{G}_3$**:** As previously, $|\Pr[\mathsf{G}_2] - \Pr[\mathsf{G}_3]| \leq \mathsf{Err}_{\mathsf{ss}}$. Since $\boldsymbol{\Psi}$ is special-sound the probability that $\mathcal{E}$ fails in extracting the witness is upper-bounded by some negligible $\varepsilon_{\mathsf{ss}}$.

In the last game, Game G_3, the environment aborts when it fails to extract the correct witness, hence the adversary $\mathcal{A}_{\mathsf{se}}$ cannot win. Thus, by the game-hoping argument,

$$|\Pr[\mathsf{G}_0] - \Pr[\mathsf{G}_3]| \leq 1 - \left(\frac{\widetilde{\mathsf{acc}}^{n_j}}{q^{n_j-1}} + \widetilde{\mathsf{acc}} \cdot \left(\frac{2^\lambda - n_j}{2^\lambda}\right)^{n_j} - \widetilde{\mathsf{acc}}\right) + \varepsilon_{\mathsf{ur}} + \varepsilon_{\mathsf{ss}} \,.$$

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

$$\frac{\widetilde{\mathsf{acc}}^{n_j}}{q^{n_j-1}} + \widetilde{\mathsf{acc}} \cdot \left(\frac{2^\lambda - n_j}{2^\lambda}\right)^{n_j} - \widetilde{\mathsf{acc}} - \varepsilon_{\mathsf{ur}} - \varepsilon_{\mathsf{ss}} \,.$$

Since $\widetilde{\mathsf{acc}}$ is probability of $\mathcal{A}_{\mathsf{se}}$ outputting acceptable transcript that does not break k-ur-ness of $\boldsymbol{\Psi}$, then $\widetilde{\mathsf{acc}} \geq \mathsf{acc} - \varepsilon_{\mathsf{ur}}$, where acc is the probability of $\mathcal{A}_{\mathsf{se}}$ outputing an acceptable proof as defined in Definition 3. It thus holds

$$\mathsf{ext} \geq \frac{(\mathsf{acc} - \varepsilon_{\mathsf{ur}})^{n_j}}{q^{n_j-1}} - \underbrace{(\mathsf{acc} - \varepsilon_{\mathsf{ur}}) \cdot \left(1 - \left(\frac{2^\lambda - n_j}{2^\lambda}\right)^{n_j}\right) - \varepsilon_{\mathsf{ur}} - \varepsilon_{\mathsf{ss}}}_{\varepsilon} \,.$$

Note that the part of Section 4 denoted by ε is negligible as $\varepsilon_{\mathsf{ur}}, \varepsilon_{\mathsf{ss}}$ are negligible, and $\left((2^\lambda - n_j)/2^\lambda\right)^{n_j}$ is overwhelming. Thus,

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

thus $\boldsymbol{\Psi}_{\mathsf{FS}}$ is simulation extractable with extraction error $\varepsilon_{\mathsf{ur}}$. □

5 Simulation extractability of $\mathbf{P}_{\mathsf{FS}}$

In this section we show that $\mathbf{P}_{\mathsf{FS}}$ is 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 2.2 and Lemma 3. This is then used to show that $\mathbf{P}$ has the 2-ur property, cf. Lemma 4.

Next, we show that $\mathbf{P}$ is computational special-sound. That is, given a number of acceptable transcripts which match on the first 3 rounds of the protocol we can either reveal a correct witness for the proven statement or use one of the transcripts to break the dlog assumption. The latter requires the AGM. More precisely, we assume that each group element included in the transcripts comes with a vector of coefficients that represents the element in the basis compound of the input group elements, i.e. $\mathbf{P}$'s CRS. See Lemma 5.

Given special-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 $\mathcal{A}_{\mathsf{se}}$ either breaks the protocol's unique response property or breaks the dlog assumption, if extracting a valid witness from a tree of transcripts is impossible. See Corollary 1.

5.1 Unique opening property of $\mathsf{PC_P}$

Lemma 3. *Let $\mathsf{PC_P}$ be a batched version of a KZG polynomial commitment [?] as described in [?] then $\mathsf{PC_P}$ has the unique opening property in the AGM.*

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 for every PPT adversary $\mathcal{A}$ the probability

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

is at most negligible.

Step 1: First, consider a case where the commitment is limited to commit to multiple polynomials which are evaluated at the same point z. 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 consider equality of the polynomials' evaluations. This holds since an adversary that manages to provide a commitment opening that holds for the real verifier, but does not hold for the idealised verifier can be used to break the dlog assumption and reveal the secret trapdoor used to produce the commitment's CRS.

For polynomials $\boldsymbol{f} = f_1, \ldots, f_k$, evaluation point z, evaluation result $\boldsymbol{s} = s_1, \ldots, s_k$, random γ, and opening $o(X)$ the idealised check verifies that

$$\sum_{i=1}^{k} \gamma^{i-1} f_i(X) - \sum_{i=1}^{k} \gamma^{i-1} s_i \equiv o(X)(X - z)\,. \tag{7}$$

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.

Step 2: Second, consider a case when the polynomials are evaluated on two points $\boldsymbol{z} = (z, z')$ and the adversary is asked to provide two openings $\boldsymbol{o} = (o, o')$. Similarly, we analyse the case of the ideal verification. In that scenario, the verifier checks whether the following equality, for γ, r' picked at random, holds:

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

Since r' has been picked at random, Eq. (8) holds while either

$$\sum_{i=1}^{k} \gamma^{i-1} \cdot f_i(X) - \sum_{i=1}^{k} \gamma^{i-1} \cdot s_i \equiv o(X)(X-z)$$

or

$$\sum_{i=1}^{k'} \gamma'^{i-1} \cdot f'_i(X) - \sum_{i=1}^{k'} \gamma'^{i-1} \cdot s'_i \equiv o'(X)(X-z')$$

does not is negligible [?]. This brings the proof back to Step 1 above. □

5.2 Unique response property

Lemma 4. *Let* $\mathsf{PC_P}$ *be commitment of knowledge with security* ε_{k}*,* $\varepsilon_{\mathsf{bind}}$*-binding and has unique opening property with security* $\varepsilon_{\mathsf{op}}$*, then probability that a* PPT *adversary* $\mathcal{A}$ *breaks* $\mathbf{P}$*'s* 2-ur *property is at most* $\varepsilon_{\mathsf{k}} + 2 \cdot \varepsilon_{\mathsf{bind}} + \varepsilon_{\mathsf{op}}$*.*

Proof (sketch). Let $\mathcal{A}(\mathbf{R}, \mathsf{crs} = ([1, \chi, \ldots, \chi^{\mathsf{n}+2}]_1, [\chi]_2))$ be an adversary trying to break the 2-ur-ness of $\mathbf{P}$. It is sufficient to observe that the first 2 rounds of the protocol determines, along with the verifiers challenges, the rest of it. In Round 3 the adversary outputs a commitment to polynomial $\mathsf{t}(X)$ which assures that all constraints of the system are fulfilled. Since the commitment scheme is deterministic, there is only one value c that is a commitment to $\mathsf{t}(X)$. Assume that $\mathcal{A}$ outputs $c' \neq c$ and is later able to open c' to $y = \mathsf{t}(\mathfrak{z})$, where $\mathfrak{z}$ is a random point determined later. Using AGM and arguments similar to [?], we argue that $\mathsf{PC_P}$ is a commitment of knowledge. That is, a AGM adversary $\mathcal{A}$ that outputs a commitment c' which it can later open, knows a polynomial f such that $[\mathsf{f}(\chi)]_1 = c'$. Thus if $c' \neq c$ and the commitment scheme is evaluation binding then $\mathcal{A}$ when picking c' picks it as a commitment to f which evaluates at $\mathfrak{z}$ to $\mathsf{t}(\mathfrak{z})$. The probability of that is negligible as there can only be degree of t many overlapping points and $\mathfrak{z}$ remains random for $\mathcal{A}$ when it computes c'. Hence, the probability that $\mathcal{A}$ is able to produce two different outputs of Round 3 is upper-bounded by $\varepsilon_{\mathsf{k}} + \varepsilon_{\mathsf{bind}}$.

In Round 4 the prover is asked to give evaluations of predefined polynomials at some point $\mathfrak{z}$. Naturally, for the given polynomials only one value at $\mathfrak{z}$ is correct. Assume $\mathcal{A}$ is able to produce two different outputs in that round: $\boldsymbol{r_4} = (\widetilde{\mathsf{a}}, \widetilde{\mathsf{b}}, \widetilde{\mathsf{c}}, \widetilde{\mathsf{S}_{\sigma 1}}, \widetilde{\mathsf{S}_{\sigma 2}}, \widetilde{\mathsf{r}}, \widetilde{\mathsf{z}})$ and $\boldsymbol{r_4} = (\widetilde{\mathsf{a}}', \widetilde{\mathsf{b}}', \widetilde{\mathsf{c}}', \widetilde{\mathsf{S}_{\sigma 1}}', \widetilde{\mathsf{S}_{\sigma 2}}', \widetilde{\mathsf{r}}', \widetilde{\mathsf{z}}')$ which suppose to be evaluations at $\mathfrak{z}$ of polynomials $\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{S}_{\sigma 1}, \mathsf{S}_{\sigma 2}, \mathsf{r}$ and an evaluation at $\mathfrak{z}\omega$ of z. Clearly, at least one of $\boldsymbol{r_4}$, $\boldsymbol{r'_4}$ has to be incorrect, thus if both evaluations are acceptable by the $\mathsf{PC}.\mathsf{Vf}$ then the evaluation binding property of PC is broken. This happens with probability upper-bounded by $\varepsilon_{\mathsf{bind}}$.

In the last round of the protocol the prover provides openings for the polynomial commitment evaluations done before. Assume $\mathcal{A}$ is able to produce two different polynomial commitment openings pairs: $\boldsymbol{r_5} = (\widetilde{\mathsf{W}_{\mathfrak{z}}}, \widetilde{\mathsf{W}_{\mathfrak{z}\omega}})$ and $\boldsymbol{r'_5} = (\widetilde{\mathsf{W}_{\mathfrak{z}}}', \widetilde{\mathsf{W}_{\mathfrak{z}\omega}}')$. Since PC has unique opening property, one of the openings has to be incorrect and should be rejected by the polynomial commitment verifier. This happens except with probability $\varepsilon_{\mathsf{op}}$

Hence, the probability that after fixing the two first rounds, the adversary is able to produce two different outputs in one of the following rounds is upper-bounded by

$$\varepsilon_{\mathsf{k}} + 2 \cdot \varepsilon_{\mathsf{bind}} + \varepsilon_{\mathsf{op}} \,.$$

□

5.3 rewinding-based knowledge soundness

Lemma 5. *Let $\mathcal{A}$ be a* PPT *algebraic adversary. The probability $\varepsilon_{\mathsf{ss}}$ that $\mathcal{A}$ breaks $(1, 1, 1, \mathsf{n} + 3, 1)$-rewinding-based knowledge soundness of* P *is upper-bounded as*

$$\varepsilon_{\mathsf{ss}} \leq \varepsilon_{\mathsf{btch}} + \varepsilon_{\mathsf{dlog}} \,,$$

where $\varepsilon_{\mathsf{btch}}$ is the (negligible) probability that P*'s idealised verification equation* $\mathsf{ve}(X)$ *accepts an invalid proof because of batching and $\varepsilon_{\mathsf{dlog}}$ is a probability that a* PPT *algorithm can break the* $(\mathsf{n} + 2, 1)$-dlog *assumption.*

Proof. Let crs be P's CRS and denote by crs_1 all CRS's $\mathbb{G}_1$-elements; that is, $\mathsf{crs}_1 = \left[1, \chi, \ldots, \chi^{\mathsf{n}+2}\right]_1$. Let $\mathcal{A}$ be an algebraic adversary that for a statement x produces a $(1, 1, 1, \mathsf{n}+3, 1)$-tree of acceptable transcripts T. Note that in all transcripts the instance x, proof elements $[\mathsf{a}(\chi), \mathsf{b}(\chi), \mathsf{c}(\chi), \mathsf{z}(\chi), \mathsf{t}(\chi)]_1$ and challenges α, β, γ are common as the transcripts share the first three rounds.

We consider two mutually disjunctive events. First, E holds when all of the transcripts are acceptable by the idealised verification equation, i.e. $\mathsf{ve}(X) = 0$, cf. Eq. (15). Second, $\overline{\mathsf{E}}$ holds when there is a transcript that is acceptable, yet for some transcript $\mathsf{ve}(\chi) = 0$, but $\mathsf{ve}(X) \neq 0$. We build a special extractor $\mathsf{Ext}_{\mathsf{ss}}$ which given the tree of transcripts T reveals the witness with an overwhelming probability when E happens. We show a reduction $\mathcal{R}_{\mathsf{dlog}}$ that, when $\overline{\mathsf{E}}$ happens, breaks the dlog assumption.

When E happens: Since the protocol P, instantiated with the idealised verification equation, is perfectly sound, except with probability of batching failure $\varepsilon_{\mathsf{btch}}$, for a valid proof π of a statement x there exists a witness w, such that $\mathbf{R}(\mathsf{x}, \mathsf{w})$ holds. Note that the polynomials $\mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X)$, which contain witness in their coefficients, have degree $(\mathsf{n} + 2)$ and since $\mathcal{A}$ answered honestly on $(\mathsf{n} + 3)$ different challenges $\mathfrak{z}$ then $(\mathsf{n} + 3)$ evaluations of these polynomials (at different points) are known. The extractor $\mathsf{Ext}_{\mathsf{ss}}$ interpolates the polynomials and reveals the corresponding witness w.

When $\overline{\mathsf{E}}$ happens: Consider a transcript that such that $\mathsf{ve}(X) \neq 0$, but $\mathsf{ve}(\chi) = 0$. Since the adversary is algebraic, all group elements included in the tree of transcripts are extended by their representation as a combination of the input $\mathbb{G}_1$-elements i.e. $\left[1, \chi, \ldots, \chi^{\mathsf{n}+2}\right]_1$. Hence all coefficients of the verification equation polynomial $\mathsf{ve}(X)$ are known and $\mathcal{R}_{\mathsf{dlog}}$ can find its zero points. Since $\mathsf{ve}(\chi) = 0$, the targeted discrete log value χ is among them. □

5.4 Honest verifier zero-knowledge

Lemma 6. Plonk *is honest verifier zero-knowledge and its simulator* Sim *does not require a CRS trapdoor.*[5] *More precisely, assume that* Plonk*'s CRS simulator* Sim_χ *produces a proof that is at most* $\varepsilon_{\mathsf{zk}}$ *far from a real proof, and* $(\mathsf{R}, \mathsf{S}, \mathsf{T}, \mathsf{f})$-*uber assumption for* $\mathsf{R}, \mathsf{S}, \mathsf{T}, \mathsf{f}$ *as defined in Eq.* (9) *is* $\varepsilon_{\mathsf{uber}}$-*secure. Then for any* PPT *adversary* $\mathcal{A}$ *its advantage in telling a proof produced by* Sim *from a real proof is upper-bounded by* $\varepsilon_{\mathsf{zk}} + \varepsilon_{\mathsf{uber}}$.

Proof. The proof goes by game-hoping. The environment that controls the games provides the adversary with a CRS crs, 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 $\mathcal{A}(\mathbf{R}, \mathsf{crs})$ picks an instance–witness pair (x, w) and gets a real proof π for it.

Game G_1: In this game for $\mathcal{A}(\mathbf{R}, \mathsf{crs})$ picks an instance–witness pair (x, w) and gets a proof π that is simulated by a simulator Sim_χ that utilises for the simulation the CRS trapdoor and proceeds as follows. In the first round the simulator Sim_χ picks randomisers $b_1, \ldots b_9$, sets w_i, for $i \in [1 \,..\, 3\mathsf{n}]$, computes polynomials $\mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X)$ and outputs $[\mathsf{a}(\chi), \mathsf{b}(\chi), \mathsf{c}(\chi)]_1$. Then it picks Round 1 challenge β, γ honestly.

In the second round Sim_χ computes the polynomial $\mathsf{z}(X)$ and outputs $[\mathsf{z}(\chi)]_1$. Then it picks randomly Round 2 challenge α.

In the third round the simulator computes polynomial $\mathsf{t}(X)$ and evaluates it at χ, then outputs $[\mathsf{t_{lo}}(\chi), \mathsf{t_{mid}}(\chi), \mathsf{t_{hi}}(\chi)]_1$. Note that this evaluation is possible only since Sim_χ knows the trapdoor.

In the last two rounds the simulator proceeds as an honest prover would proceed and picks corresponding challenges at random as an honest verifier would.

$\mathsf{G}_0 \mapsto \mathsf{G}_1$**:** Since Plonk is zero-knowledge, probability that $\mathcal{A}$ outputs a different bit in both games is negligible. Hence

$$|\Pr[\mathsf{G}_0] - \Pr[\mathsf{G}_1]| \le \varepsilon_{\mathsf{zk}}.$$

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

[5] 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 CRS simulator.

Since the simulator Sim does not know a witness w for the proven statement x, it cannot compute the output of Round 1 accordingly to the protocol. Instead, it picks randomly both the randomisers $b_1, \ldots, b_6$ and sets $\mathsf{w}_i = 0$ for $i \in [1 \mathinner{..} 3\mathsf{n}]$. 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 the second round, the simulator cannot the simulator computes z from the newly picked randomisers b_7, b_8, b_9 and coefficients of polynomials $\mathsf{a}, \mathsf{b}, \mathsf{c}$. Then it evaluates z 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.

The next round starts by the simulator 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 $\mathsf{a}(\mathfrak{z}), \mathsf{b}(\mathfrak{z}), \mathsf{c}(\mathfrak{z}), \mathsf{S}_{\sigma 1}(\mathfrak{z}), \mathsf{S}_{\sigma 2}(\mathfrak{z}), \mathsf{PI}(\mathfrak{z}), \mathsf{L}_1(\mathfrak{z}), \mathsf{Z_H}(\mathfrak{z})$, $\mathsf{z}(\mathfrak{z}\omega)$ and computes $\mathsf{t}(X)$ honestly. Since for a random $\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{z}$ the constraint system is (with overwhelming probability) not satisfied and the constraints-related polynomials are not divisible by $\mathsf{Z_H}$, $\mathsf{t}(X)$ is a rational function, not a polynomial. Then, the simulator evaluates $\mathsf{t}(X)$ at $\mathfrak{z}$ and picks randomly a degree-$(\mathsf{n}+2)$ polynomial $\tilde{\mathsf{t}}(X)$ such that $\mathsf{t}(\mathfrak{z}) = \tilde{\mathsf{t}}(\mathfrak{z})$ and publishes a commitment $[\tilde{\mathsf{t}}(\chi)]_1$. 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. Supp. Mat. E 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.

$\mathsf{G}_1 \mapsto \mathsf{G}_2$: We now describe the reduction $\mathcal{R}$ which relies on the $(\mathsf{R}, \mathsf{S}, \mathsf{T}, \mathsf{f})$-uber assumption where $\mathsf{R}, \mathsf{S}, \mathsf{T}, \mathsf{f}$ are polynomials over variables $\boldsymbol{B} = B_1, \ldots B_9$ and are defined as follows. Let $E = \{\{1, 2\}, \{3, 4\}, \{5, 6\}, \{7, 8, 9\}\}$. Let

$$\begin{aligned}\mathsf{R}(\boldsymbol{B}) = {} & \{B_i \mid i \in A,\ A \in E\} \cup \{B_i B_j \mid i \in A, j \in B,\ A \neq B,\ A, B \in E\} \cup \\ & \{B_i B_j B_k \mid i \in A,\ j \in B,\ k \in C,\ A \neq B \neq C \neq A,\ A, B, C \in E\} \cup \\ & \{B_i B_j B_k B_l \mid i \in A,\ j \in B,\ k \in C,\ l \in D,\ A, B, C, D \text{ all different and in } E\} \\ & \setminus \{B_1 B_3 B_5 B_7\}\,, \\ \mathsf{S}(\boldsymbol{B}) = {} & \emptyset\,, \\ \mathsf{T}(\boldsymbol{B}) = {} & \emptyset\,, \\ \mathsf{f}(\boldsymbol{B}) = {} & B_1 B_3 B_5 B_7\,.\end{aligned} \tag{9}$$

That is, the elements of R are all singletons, pairs and triplets of B_i variables that occur in polynomial $\mathsf{t}(\boldsymbol{B})$ except the challenge element $\mathsf{f}(\boldsymbol{B}) = B_1 B_3 B_5 B_7$. 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 $[w]_1$ where w is either $\mathsf{f}(\boldsymbol{b}) = b_1 b_3 b_5 b_7$ or a random value. Then it picks χ, $\mathfrak{z}$ and computes the CRS crs 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

$$\left[\tilde{\mathsf{b}}_i\right]_1(X) = (X - \mathfrak{z})(X - \omega\mathfrak{z})\left[b_i\right]_1(X) + \xi_i(X - \mathfrak{z})\left[1\right]_1 + \zeta_i(X - \omega\mathfrak{z})\left[1\right]_1, \leftarrow\!\$\ \mathbb{F}_p$$

for $i \in [1\,..\,9]$ and $\xi_i, \zeta_i \leftarrow\!\$\ \mathbb{F}_p$. 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 $\left[B_iB_j, B_iB_jB_kB_l\right]_1 \in \mathsf{R}$. This is possible since $\mathcal{R}$ knows all singletons $\left[b_1, \ldots, b_9\right]_1$ and pairs $\left[b_ib_j\right]_1 \in \mathsf{R}$ which can be used to compute all required pairs $\left[\tilde{b}_i\tilde{b}_j\right]_1$:

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

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.

For the challenge $\left[w\right]_1$ $\mathcal{R}$ sets $\left[\tilde{w}\right]_1 = \left[\tilde{b}_1\tilde{b}_3\tilde{b}_5\tilde{b}_7\right]_1$, where $\left[b_1b_3b_5b_7\right]_1$ is substituted by $\left[w\right]_1$. Next it runs the adversary $\mathcal{A}(\mathbf{R}, \mathsf{crs})$ and obtains from $\mathcal{A}$ an instance–witness pair (x, w). $\mathcal{R}$ now prepares a simulated proof.

Round 1 $\mathcal{R}$ computes $\left[\mathsf{a}(\chi)\right]_1$ 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 $\left[\mathsf{b}(\chi)\right]_1, \left[\mathsf{c}(\chi)\right]_1$. $\mathcal{R}$ publishes the obtained values and picks a Round 1 challenge β, γ at random.

Round 2 $\mathcal{R}$ computes $\left[\mathsf{z}(\chi)\right]_1$ using $\tilde{b}_7, \tilde{b}_8, \tilde{b}_9$ and publishes it. Then it picks randomly the challenge α.

Round 3 The reduction computes $\left[\mathsf{t}(\chi)\right]_1$ using $\tilde{w}$ as it was equal $\tilde{b}_1\tilde{b}_3\tilde{b}_5\tilde{b}_7$. That is, if $w = b_1b_3b_5b_7$ then $\mathsf{t}(\chi)$ is as computed by the simulator Sim_χ, otherwise, if w is random then $\mathsf{t}(\chi)$ is random as well, thus it is computed as Sim would compute. The reduction computes and outputs $\left[\mathsf{t}_{\mathsf{lo}}(\chi), \mathsf{t}_{\mathsf{mid}}(\chi), \mathsf{t}_{\mathsf{hi}}(\chi)\right]_1$ such that $\mathsf{t}(X) = \mathsf{t}_{\mathsf{lo}}(X) + X^{\mathsf{n}}\mathsf{t}_{\mathsf{mid}}(X) + X^{2\mathsf{n}}\mathsf{t}_{\mathsf{hi}}(X)$. 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.

Is the adversary $\mathcal{A}$'s output distribution differ in Game G_1 and G_2 then the reduction uses it to distinguish between $w = b_1b_3b_5b_7$ and w being random, thus $|\Pr[\mathsf{G}_1] - \Pr[\mathsf{G}_2]| \leq \varepsilon_{\mathsf{uber}}$. Eventually, $|\Pr[\mathsf{G}_0] - \Pr[\mathsf{G}_2]| \leq \varepsilon_{\mathsf{zk}} + \varepsilon_{\mathsf{uber}}$. □

5.5 From special-soundness and unique response property to simulation extractability of P_{FS}

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

Corollary 1 (Simulation extractability of P_{FS}). *Assume that P is 2-ur with security $\varepsilon_{\mathsf{ur}}(\lambda)$, and computational special-sound with security $\varepsilon_{\mathsf{ss}}(\lambda)$. Let $\mathcal{H}\colon \{0,1\}^* \to \{0,1\}^\lambda$ be a random oracle. Let $\mathcal{A}_{\mathsf{se}}$ be a PPT adversary that can make up to q random oracle queries and outputs an acceptable proof for P_{FS} with probability at least acc. Then P_{FS} is simulation-extractable with extraction error $\eta = \varepsilon_{\mathsf{ur}}$. The extraction probability ext is at least*

$$\mathsf{ext} \geq \frac{1}{q^{\mathsf{n}+2}} (\mathsf{acc} - \varepsilon_{\mathsf{ur}})^{\mathsf{n}+3} - \varepsilon\,,$$

for some negligible ε and n being the number of contrains in the proven circuit.

6 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 $(n_1, \ldots, n_\mu)$-special soundness holds for 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 CRS-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.

Supplementary Material

A Additional preliminaries

A.1 Computational assumptions

Definition 7 ((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*

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

Definition 8 ((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*

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

BBG uber assumption The original (decisional) uber assumption by Boneh et al. states that for type-III pairings:

Definition 9 ($(\mathsf{R}, \mathsf{S}, \mathsf{T}, \mathsf{f})$-uber assumption [?]). *Let $\mathsf{R}, \mathsf{S}, \mathsf{T}$ be defined as above, $(x', x_1, \ldots, x_c) \leftarrow\!\!\$\ \mathbb{F}_p^{c+1}$ and let f be independent on $(\mathsf{R}, \mathsf{S}, \mathsf{T})$, cf. Definition 1. Then, for any PPT adversary $\mathcal{A}$*

$$\Pr[\mathcal{A}([\mathsf{R}(x_1, \ldots x_c)]_1, [\mathsf{S}(x_1, \ldots, x_c)]_2, [\mathsf{T}(x_1, \ldots, x_c)]_T, [\mathsf{f}(x_1, \ldots, x_c)]_T) = 1] \approx_\lambda$$
$$\Pr[\mathcal{A}([\mathsf{R}(x_1, \ldots x_c)]_1, [\mathsf{S}(x_1, \ldots, x_c)]_2, [\mathsf{T}(x_1, \ldots, x_c)]_T, [x']_T) = 1].$$

A.2 Polynomial commitment scheme

Polynomial commitment schemes used in this paper has been presented in Fig. 1, for Plonk's PC, and in Fig. 3, for Sonic's PC.

B Simulation-extractability of sigma protocols and forking lemma

Theorem 2 (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 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* [?].

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

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

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

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

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

```
F_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 $\mathsf{F}_{\mathcal{Z}}$

Lemma 7 (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 $i \in [0 \mathinner{..} q]$ and s is called a* side output*. Denote by* IG *a randomised instance generator. We denote by* acc *the probability*

$$\Pr[i > 0 \mid y \leftarrow \mathsf{IG}; h_1, \ldots, h_q \leftarrow_\$ H; (i, s) \leftarrow \mathcal{Z}(y, h_1, \ldots, h_q)] .$$

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*

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

In case of a NIZK obtained by applying the Fiat–Shamir transform to a sigma protocol, forking lemma could be used as follows. Let Ext be an extractor tasked to reveal the witness from $\mathcal{A}$ who provides an acceptable proofs for a picked statement. Ext gets as input the adversary's code and its randomness, this is also the input y passed to the forking algorithm F. Algorithm $\mathcal{Z}$ runs the code of the adversary on a given random oracle responses $h_1, \ldots, h_q$ and returns adversary's state s, which contains an instance x and its proof π, and index i of the random oracle query such that h_i was random oracle's response to query (x, a) used in π. Then the first $i-1$ random oracle responses remain the same and fresh $h'_i, \ldots, h'_q$. The adversary is run again on the same randomness r and it sees the same random oracle responses up to the index $i-1$. Thus, it behaves exactly the same as in the previous run (up to the point). It is then presented with a new random oracle responses and finally outputs a new state s'. We also require that the adversary uses the query of the same index i to ask about $\mathcal{H}(\mathsf{x}, a)$ what is needed for the security reduction, cf. [?].

C Simulation extractability of S_{FS}

C.1 Unique opening property of PC_{S}

Lemma 8. *Let PC_{S} be a batched version of a KZG polynomial commitment [?] as described in [?] then 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, $c \in \mathbb{G}$ be the commitment, $s \in \mathbb{F}_p$ the evaluation, and $o \in \mathbb{F}_p$ be the commitment opening. We need to show that for every PPT adversary $\mathcal{A}$ probability

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

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 consider equality of the polynomials' evaluations.

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

$$\alpha(X^{\mathsf{d}-\mathsf{max}} f(X) \cdot X^{-\mathsf{d}+\mathsf{max}} - s) \equiv \alpha \cdot o(X)(X - z)\,, \tag{10}$$

what is equivalent to

$$f(X) - s \equiv o(X)(X - z)\,. \tag{11}$$

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.

□

C.2 Unique response property

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

Lemma 9. *If a polynomial commitment scheme* $\mathsf{PC_S}$ *is evaluation binding with parameter* $\varepsilon_{\mathsf{bind}}$ *and has unique openings property with parameter* $\varepsilon_{\mathsf{op}}$*, then* S *is* $1\text{-}\mathsf{ur}$ *with parameter* $\varepsilon_{\mathsf{ur}} \le \varepsilon_{\mathsf{bind}} + \varepsilon_{\mathsf{op}}$*.*

Proof. Let $\mathcal{A}$ be an adversary that breaks $1\text{-}\mathsf{ur}$-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 $\mathcal{A}$ breaks the evaluation binding property of $\mathsf{PC_S}$, while for the second case we show that it breaks the unique opening property of $\mathsf{PC_S}$.

The proof goes similarly to the proof of Lemma 4 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 some randomly picked points, or shows that the evaluations were performed correctly. Naturally 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 $\mathsf{PC_S}$'s evaluation binding would be broken. Alternatively, if $\mathcal{A}$ was able to provide two openings W and $\tilde{\mathsf{W}}$ for $y = \mathsf{x}$ then the unique opening property would be broken.

Hence the probability that $\mathcal{A}$ breaks $1\text{-}\mathsf{ur}$-property of $\mathsf{PC_S}$ is upper-bounded by $\varepsilon_{\mathsf{bind}} + \varepsilon_{\mathsf{op}}$. □

C.3 rewinding-based knowledge soundness

Lemma 10. *Let* $\mathcal{A}$ *be a* PPT *algebraic adversary. The probability* $\varepsilon_{\mathsf{ss}}$ *that* $\mathcal{A}$ *breaks rewinding-based knowledge soundness of* S *is upper-bounded as*

$$\varepsilon_{\mathsf{ss}} \le \varepsilon_{\mathsf{s}} + \varepsilon_{\mathsf{ldlog}} ,$$

where ε_{s} *is a soundness error of the protocol, and* $\varepsilon_{\mathsf{ldlog}}$ *is a probability that a* PPT *algorithm can break the* $(\mathsf{d}, \mathsf{d})\text{-}\mathsf{ldlog}$ *assumption.*

Proof. The proof goes similarly to the proof of Lemma 5. Let $\mathcal{A}$ be an adversary that produces a $(1, 1, \mathsf{n} + 1, 1, 1)$-tree of acceptable transcripts T for a statement x. We consider two disjunctive events E and $\overline{\mathsf{E}}$. The first corresponds to a case when all transcripts in T are acceptable for the ideal verifier, i.e. $\mathbf{\mathsf{ve}}(X) = \mathbf{0}$. In that case we show an extractor $\mathsf{Ext_{ss}}$ that from T extracts a valid witness w. The second, corresponds to a case when T contains a transcript that is acceptable by the real verifier but is not acceptable by the ideal verifier. In that case we show a reduction $\mathcal{R}_{\mathsf{ldlog}}$ that uses $\mathcal{A}$ to break the $(\mathsf{d}, \mathsf{d})\text{-}\mathsf{ldlog}$ assumption with at least the same probability.

When E happens: Since S is perfectly sound regarding the ideal verifier, for an acceptable proof π for a statement x there exists a witness w such that $\mathbf{R}(\mathsf{x}, \mathsf{w})$

holds and the polynomial $\mathsf{r}(X, Y)$ contains witness at its coefficients. Note that the witness-carrying polynomial $\mathsf{r}(X, y)$ has degree at most (n) and since $\mathcal{A}$ answered correctly on $(\mathsf{n} + 1)$ different challenges z (for the sake of concreteness let us call them $z_1, \ldots, z_{\mathsf{n}+1}$) then $(\mathsf{n}+\mathsf{Q}+1)$ evaluations $\mathsf{r}(z_1, y), \ldots, \mathsf{r}(z_{\mathsf{n}+\mathsf{Q}+1}, y)$ of $\mathsf{r}(X, y)$ are known. The extractor $\mathsf{Ext}_{\mathsf{ss}}$ interpolates $\mathsf{r}(X, y)$ and reveals the corresponding witness w.

When $\overline{\mathsf{E}}$ happens: Consider a transcript such that for some verification equation $\mathsf{ve}_i(X) \neq 0$, but $\mathsf{ve}_i(\chi) = 0$. Since the adversary is algebraic, all group elements included in the tree of transcripts are extended by their representation as a combination of the input $\mathbb{G}_1$ or $\mathbb{G}_2$-elements. Hence all coefficients of the verification equation polynomial $\mathsf{ve}_i(X)$ are known and $\mathcal{R}_{\mathsf{dlog}}$ can find its zero points. Since $\mathsf{ve}_i(\chi) = 0$, the targeted discrete log value χ is among them. □

C.4 Honest verifier zero-knowledge

Lemma 11. Sonic *is honest verifier zero-knowledge.*

Proof. The simulator proceeds as follows. In the first round, it picks randomly vectors $\boldsymbol{a}$, $\boldsymbol{b}$ and sets

$$\boldsymbol{c} = \boldsymbol{a} \cdot \boldsymbol{b}. \tag{12}$$

Then it pick randomisers $c_{\mathsf{n}+1}, \ldots, c_{\mathsf{n}+4}$, honestly computes polynomials $\mathsf{r}(X, Y), \mathsf{r}'(X, Y), \mathsf{s}(X, Y)$ and $\mathsf{t}(X, Y)$ and concludes the first round as an honest prover would. Because of the randomisers polynomial $\mathsf{r}(X, Y)$ computed by the simulator is indistinguishable from a polynomial provided by an honest user for a distinguisher that only learns $\mathsf{R}, \mathsf{T}, \mathsf{a}, \mathsf{b}$ from the proof (the other proof elements are fixed by these elements and the public information). See Supp. Mat. F.1.

Next, Sim computes the first verifier's challenge y such that $\mathsf{t}(X, y)$ is a polynomial that has 0 as a coefficient next to X^0. I.e. $\mathsf{t}(0, y) = 0$. As noted in [?], the coefficient next to X^0 in $\mathsf{t}(X, Y)$ equals

$$\mathsf{t}(0, Y) = \boldsymbol{a} \cdot \mathbf{u}(Y) + \boldsymbol{b} \cdot \mathbf{v}(Y) + \boldsymbol{c} \cdot \mathbf{w}(Y) + \sum_{i=1}^{\mathsf{n}} a_i b_i (Y^i + Y^{-i}) - \mathsf{k}(Y), \tag{13}$$

for public $\mathbf{u}(Y), \mathbf{v}(Y), \mathbf{w}(Y), \mathsf{k}(Y)$ as defined in Supp. Mat. F.1 (Vectors $\boldsymbol{u_q}, \boldsymbol{v_q}, \boldsymbol{w_q}$ are n-elements long and correspond to the Q linear constrains of the proof system. Field element k_q is the instance value). When the proven instance is correct, $\mathsf{t}(0, Y)$ is a zero polynomial. See [?] for details. Also, when Eq. (12) holds, that polynomial simplifies to

$$t(0, Y) = \boldsymbol{a} \cdot \mathbf{u}(Y) + \boldsymbol{b} \cdot \mathbf{v}(Y) + \boldsymbol{c} \cdot \tilde{\mathbf{w}}(Y) - \mathsf{k}(Y), \tag{14}$$

where $\tilde{\mathbf{w}}(Y)$ is defined as

$$\tilde{\mathsf{w}}_{\mathsf{i}}(Y) = \sum_{q=1}^{\mathsf{Q}} Y^{q+\mathsf{n}} w_{q,i}\,.$$

Note that the polynomial from Eq. (14) is a "classical", i.e. non-Laurent, polynomial. Also, $t(0, Y)$ is a polynomial of degree $(\mathsf{Q} + \mathsf{n})$ with 0 coefficients for Y^i, for $i \in [0 \mathinner{..} \mathsf{n}]$. Also, since $\boldsymbol{a}, \boldsymbol{b}$ were picked at random, $\mathsf{t}(0, Y)/Y^{\mathsf{n}+1}$ is a degree-$(\mathsf{Q} - 1)$ polynomial of random coefficients. Recall, that the view of the adversary is independent of $\boldsymbol{a}, \boldsymbol{b}$ because of randomisers $c_{\mathsf{n}+1}, \ldots, c_{\mathsf{n}+4}$.

The probability that a random degree-$(\mathsf{Q} - 1)$ polynomial over $\mathbb{F}_p[Y]$ has a root is at least $1/(\mathsf{Q} - 1)!$, see Lemma 14 for a proof of that bound. Since we assume that $\mathsf{Q} = \mathsf{poly}(\lambda)$, we can say that the polynomial $\mathsf{t}(0, Y)$ as computed by the simulator has roots with non-negligible probability. Furthermore, these roots can be found and the simulator picks fresh $\boldsymbol{a}, \boldsymbol{b}$ until $\mathsf{t}(0, Y)$ has a root. As the roots of a random polynomial are themselves random $\mathbb{F}_p$ element, the challenge y picked by the simulator comes from the same distribution as if it was picked by an honest verifier.

The simulator continues building the transcript by honestly computing the prover's messages and by picking verifier's challenges at random. This and the fact that $\mathsf{t}(0, y) = 0$ guarantees that the transcript provided by the simulator is acceptable and comes from the same distribution as a transcript of an honest prover and verifier. □

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 CRS trapdoor from a CRS-generator. Unfortunately, a NIZK made subversion zero-knowledge by this approach cannot achieve perfect Sub-ZK as one has to count in the probability of extraction failure. However, with the simulation presented in Lemma 11, the trapdoor is not required for the simulator as it is able to simulate the execution of the protocol just by picking appropriate (honest) verifier's challenges. This result transfers to S_{FS}, where the simulator can program the random oracle to provide challenges that fits it.

C.5 From special-soundness and unique response property to simulation extractability of S_{FS}

Since Lemmas 9 and 10 hold, S is 1-ur and computationally special sound. We now make use of Theorem 1 and show that S_{FS} is simulation-extractable as defined in Definition 3.

Corollary 2 (Simulation extractability of S_{FS}). *Assume that S is 1-ur with security $\varepsilon_{\mathsf{ur}}(\lambda)$, and computational special-sound with security $\varepsilon_{\mathsf{ss}}(\lambda)$. Let $\mathcal{H}\colon \{0,1\}^* \to \{0,1\}^\lambda$ be a random oracle. Let $\mathcal{A}_{\mathsf{se}}$ be a PPT adversary that can make up to q random oracle queries and outputs an acceptable proof for S_{FS} with probability at least acc. Then S_{FS} is simulation-extractable with extraction error $\eta = \varepsilon_{\mathsf{ur}}$. The extraction probability ext is at least*

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

for some negligible ε.,

D Omitted lemmas and proofs

Lemma 12. *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*

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

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 $|\mathbb{E}\left[X\right]| \leq \infty$ and f is a Borel convex function then

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

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

Lemma 13 (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*

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

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

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

Lemma 14. *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

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

□

E Plonk protocol rolled out

Plonk CRS generating algorithm $\mathsf{KGen}(\mathbf{R})$: For the sake of simplicity of security reductions presented in this paper, we describe here a simplified CRS generating algorithm which produces only these CRS elements that cannot be computed without knowing the secret trapdoor χ. The rest of the preprocessed input, as it is called in [?], can be computed using these CRS elements thus we leave them to be computed by the prover, verifier, and simulator.

The CRS generating algorithm picks at random $\chi \leftarrow\$ \mathbb{F}_p$, computes and outputs

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

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

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

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

$$\beta = \mathcal{H}(\mathsf{trans}, 0), \qquad \gamma = \mathcal{H}(\mathsf{trans}, 1).$$

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

Round 3 Get the challenge $\alpha = \mathcal{H}(\mathsf{trans})$, 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 $\mathfrak{z} \in \mathbb{F}_p$, $\mathfrak{z} = \mathcal{H}(\mathsf{trans})$. 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

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

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 $v \in \mathbb{F}_p$, $v = \mathcal{H}(\mathsf{trans})$. Compute the openings for the polynomial commitment scheme

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

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

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

Plonk **verifier** $\mathsf{V}(\mathbf{R}, \mathsf{crs}, \mathsf{x}, \pi)$**:**
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 $\beta, \gamma, \alpha, \alpha', \mathfrak{z}, v, u$ from the transcript.
Step 5 Compute zero polynomial evaluation $\mathsf{Z_H}(\mathfrak{z}) = \mathfrak{z}^{\mathsf{n}} - 1$.
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

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

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

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

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

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

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

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

Step 12 Check whether the verification $\mathsf{ve}(\chi)$ equation holds

$$\left([\mathsf{W}_{\mathfrak{z}}(\chi)]_1 + u \cdot [\mathsf{W}_{\mathfrak{z}\omega}(\chi)]_1\right) \bullet [\chi]_2 - \\ \left(\mathfrak{z} \cdot [\mathsf{W}_{\mathfrak{z}}(\chi)]_1 + u\mathfrak{z}\omega \cdot [\mathsf{W}_{\mathfrak{z}\omega}(\chi)]_1 + [F]_1 - [E]_1\right) \bullet [1]_2 = 0\,. \quad (15)$$

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).

Since the original paper [?] lacks of explanation how the simulator of Plonk works, it is presented here.

F Sonic protocol rolled out

F.1 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

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

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

$$\begin{aligned} \mathsf{u_i}(Y) &= \sum_{q=1}^{\mathsf{Q}} Y^{q+\mathsf{n}} u_{q,i}\,, & \mathsf{w_i}(Y) &= -Y^i - Y^{-i} + \sum_{q=1}^{\mathsf{Q}} Y^{q+\mathsf{n}} w_{q,i}\,, \\ \mathsf{v_i}(Y) &= \sum_{q=1}^{\mathsf{Q}} Y^{q+\mathsf{n}} v_{q,i}\,, & \mathsf{k}(Y) &= \sum_{q=1}^{\mathsf{Q}} Y^{q+\mathsf{n}} k_q. \end{aligned} \tag{16}$$

In Sonic we will use commitments to the following polynomials.

$$\begin{aligned} \mathsf{r}(X,Y) &= \sum_{i=1}^{\mathsf{n}} \left(a_i X^i Y^i + b_i X^{-i} Y^{-i} + c_i X^{-i-\mathsf{n}} Y^{-i-\mathsf{n}}\right) \\ \mathsf{s}(X,Y) &= \sum_{i=1}^{\mathsf{n}} \left(u_i(Y) X^{-i} + v_i(Y) X^i + w_i(Y) X^{i+\mathsf{n}}\right) \\ \mathsf{r}'(X,Y) &= \mathsf{r}(X,Y) + \mathsf{s}(X,Y) \\ \mathsf{t}(X,Y) &= \mathsf{r}(X,1)\mathsf{r}'(X,Y) - \mathsf{k}(Y)\,. \end{aligned}$$

F.2 Algorithms rolled out

Sonic CRS generation $\mathsf{KGen}(\mathbf{R})$: The CRS generating algorithm picks randomly $\alpha, \chi \leftarrow\!\!\$\, \mathbb{F}_p$ and outputs

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

Sonic prover $\mathsf{P}(\mathbf{R}, \mathsf{crs}, \mathsf{x}, \mathsf{w} = \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$:

Round 1 The prover picks randomly randomisers $c_{\mathsf{n}+1}, c_{\mathsf{n}+2}, c_{\mathsf{n}+3}, c_{\mathsf{n}+4} \leftarrow\!\!\$\, \mathbb{F}_p$. Set $\mathsf{r}(X,Y) \leftarrow \mathsf{r}(X,Y) + \sum_{i=1}^{4} c_{\mathsf{n}+i} X^{-2\mathsf{n}-i}$. Commit to $\mathsf{r}(X,1)$ by setting and outputting $R \leftarrow \mathsf{Com}(\mathsf{crs}, \mathsf{n}, \mathsf{r}(X,1))$. Then it gets challenge y from the verifier.

Round 2 P commits to $\mathsf{t}(X,y)$ by setting and outputting $T \leftarrow \mathsf{Com}(\mathsf{crs}, \mathsf{d}, \mathsf{t}(X,y))$. Then it gets a challenge z from the verifier.

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

$$W_a = \mathsf{Op}(\mathsf{crs}, z, \mathsf{r}(z, 1), \mathsf{r}(X, 1))$$
$$W_b = \mathsf{Op}(\mathsf{crs}, yz, \mathsf{r}(yz, 1), \mathsf{r}(X, 1))$$
$$W_t = \mathsf{Op}(\mathsf{crs}, z, \mathsf{t}(z, y), \mathsf{t}(X, y))$$

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. Then it obtains a challenge u from the verifier.

Round 4 In the next round the prover computes $C \leftarrow \mathsf{Com}(\mathsf{crs}, \mathsf{d}, \mathsf{s}(u, x)) \cdot [\mathsf{s}(u, x)]_1$ and compute commitments' openings

$$W = \mathsf{Op}(\mathsf{crs}, u, \mathsf{s}(X, y)),$$
$$Q = \mathsf{Op}(\mathsf{crs}, y, \mathsf{s}(u, Y)),$$

and returns $W, Q, \mathsf{s}(u, y)$. Eventually the prover gets the last challenge from the verifier—z.

Round 5 In the final round, P computes opening $Q_z = \mathsf{Op}(\mathsf{crs}, z, \mathsf{s}(u, X))$ and outputs Q_z and $\mathsf{s}(u, z)$.

Sonic **verifier** $\mathsf{V}(\mathbf{R}, \mathsf{crs}, \mathsf{x}, \pi)$**:** 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:

$$\begin{array}{ll} \mathbf{PC_S}.\mathsf{V}(\mathsf{crs}, \mathsf{n}, R, z, a, W_a), & \mathbf{PC_S}.\mathsf{V}(\mathsf{crs}, \mathsf{d}, S, u, s, W), \\ \mathbf{PC_S}.\mathsf{V}(\mathsf{crs}, \mathsf{n}, R, yz, b, W_b), & \mathbf{PC_S}.\mathsf{V}(\mathsf{crs}, \mathsf{d}, C, y, s, Q), \\ \mathbf{PC_S}.\mathsf{V}(\mathsf{crs}, \mathsf{d}, T, z, t, W_t), & \mathbf{PC_S}.\mathsf{V}(\mathsf{crs}, \mathsf{d}, C, z, s_z, Q_z), \end{array}$$

and accepts the proof iff all the checks holds.