On Simulation-Extractability of Universal zkSNARKs

Markulf Kohlweiss[1,2] and Michał Zając[3]

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

Abstract. 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 simulation-extractable when made non-interactive by the Fiat–Shamir transform. We prove that two efficient updatable universal zkSNARKs—Plonk [27] and Sonic [40]—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

In recent years we have seen indisputable progress in building efficient zero-knowledge proof systems, e.g. [14, 19, 28, 31, 32, 37, 38, 41] 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 (ideally constant-length) proofs. Succinctness makes zkSNARKs especially useful for deployment in real systems, e.g. [4, 17, 18, 43, 48]. This in turn raises the question: *Are the security models of current zkSNARKs adequate for real-life deployment?* This question is particularly pertinent when one realises that all zkSNARKs with constant-sized proofs are shown secure in the structured reference string (SRS) model where the SRS 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—necessarily 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 secure real-life deployment of zkSNARKs require answers to questions such as—*How can the parties be sure that the trapdoor has not leaked? Who is the party that generates the SRS? Can this party be trusted?* 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 SRS may be unrealistic and subvert the whole purpose of a decentralised system.

Universal, updateable, and subversion-sound SNARKs. One of the first to tackled this question were Bellare et al. [6] who proposed notions of *subversion zero knowledge* and *subversion soundness* and in turn Groth16 [32], a popular zkSNARK employed in industry was shown to be subversion zero-knowledge [1, 24]. Although efficient Groth16 comes with a drawback—the SRS 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 SRS-s. Since SRS 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 SRS for all circuits of a given size are called *universal*. A constant size universal zkSNARK was proposed by Groth et al. [33]. This particular proof system also introduced a novel security property called *updatable soundness*. Because of the Bellare et al. [6] 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 SRS, i.e. to take a SRS and modify it in a well-defined and verifiable way to obtain a new SRS. Updates guarantee that if at least one of the SRS-updating parties is honest then the proof system is (knowledge) sound. Although inefficient, as the SRS length is quadratic to the size of the proven statements, [33] set a new paradigm for designing zkSNARKs.

The first universal zkSNARK with updatable and short SRS was Sonic proposed by Maller et al. in [40]. Eventually, Gabizon et al. designed Plonk [27] which currently is the most efficient updatable universal zkSNARK. Independently, Chiesa et al. [16] proposed Marlin with efficiency comparable 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, the constant proof size, and the universality of the SRS, as well as for their security model that diminishes the need for a trusted party to provide a SRS.

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

State of the art—simulation-extractable updatable universal zkSNARKs. To the best of our knowledge, there are zkSNARKs that are simulation-extractable [3,12,32,34] and zkSNARKs that are universal [16, 27, 33, 40], however there are no known zkSNARKs that enjoy both of these properties out-of-the-box (even for a weaker notion of simulation extractability). Obviously, given a universal zkSNARK one could lift it to be simulation-extractable using techniques described e.g. in [2, 36], but such a lift comes with inevitable 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 [2] preserves updatability).

On the popularity and power of Fiat-Shamir. The Fiat-Shamir heuristic takes a public-coin interactive protocol and makes it interactive by computing the verifier's public coins by hashing the current protocol transcript. While in principle justifiable in the Random Oracle model [8], it is theoretically unsound [29] and so should be used with care. Nevertheless, the Fiat-Shamir heuristic is a popular design tool when it comes to constructing zkSNARKs. Many works, including [16,27,40], design interactive protocols and prove them secure. However, they then only conjecture the security for their non-interactive variants by employing the Fiat-Shamir heuristic. The more rigorous approach is to prove security of the Fiat-Shamir heuristic in the Random Oracle model.

1.1 Our contribution

We show that a class of computationally special-sound interactive proofs of knowledge that are zero-knowledge in the standard model and have a unique response property *are simulation-extractable out-of-the box* in the Random Oracle model when the Fiat–Shamir transformation is applied to them. Although a similar problem has been already tackled by Faust et al. [22] for special-sound 3-message Σ-protocols, we extend it to a much wider class of protocols and thus make it applicable to zkSNARKs.

We follow Faust et al. in their definition of simulation extractability. They called their simulation-extractability *weak* which relates to the fact that the extractor's probability of returning a witness depends on the adversary's probability acc of producing an acceptable proof. More precisely, the extractor is not guaranteed to succeed if the adversary outputs an acceptable proof with probability acc smaller than knowledge error ν. On the other hand, we show that this ν can be arbitrarily small, even negligible. However, for acc close to ν the extractor becomes fairly inefficient. On the other hand, the extractor considered in this paper is black-box as it only needs an oracle access to the adversary. Importantly, it does not depend on any knowledge assumption. (However, it also depends on a number of properties required from the protocol. In case of Plonk and Sonic, these properties are often proven in the AGM.) We also note that when an extractor depends on rewinding the adversary, it is expected to have extractor's success probability dependent on the adversary's probability of returning a valid proof.

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. To obtain this result, without

having to change anything at all in these protocols, we define new intermediary properties satisfied by these multi-round computationally sound protocols and their building blocks: computational special soundness, generalized unique response, and a generalized forking lemma. These conceptual insights into interactive SNARK systems help us overcome a number of challenges, as we explain below.

1.2 Our techniques

Before we continue, we note that Plonk and Sonic—as originally presented in [27] and [40]—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 [22], 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 is not enough. 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. Concretely, $(\mathsf{n}+3)$—where n is the number of constraints in the proven circuit—for $\mathbf{P}$ and $(\mathsf{n}+\mathsf{Q}+1)$—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 structured 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—even those that contain an acceptable proof for an incorrect statement. In this paper we define a weaker version of special soundness—a computational special soundness. That is, we show that it is possible to extract a witness from all but negligibly many trees of acceptable transcripts produced by probabilistic polynomial time (PPT) adversaries. Said otherwise, an adversary that produced a tree of acceptable transcripts from which one cannot extract can be used 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 [23], that except for the first message, all messages sent by the prover are deterministic (intuitively, the prover can only employ fresh randomness in the first message of the protocol). Again, we can not use this definition right out of the box. $\mathbf{P}$ does not satisfy it—both the 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 starting from its i-th message. For our proof it is sufficient that this property is met by $\mathbf{P}$ for $i = 3$.

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 [35] proposed in [27] and [40] 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 but acceptable openings of the commitment.

HVZK. In order to show our result we also show that (interactive) $\mathbf{P}$ and $\mathbf{S}$ 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 SRS 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 reordering the 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 $\mathbf{P}_{\mathsf{FS}}$ and $\mathbf{S}_{\mathsf{FS}}$ 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 an adversary who provides two proofs for the same statement that share the first message. Since $\mathbf{P}$ and $\mathbf{S}$ have more than 3 messages and are not special-sound, the forking lemma, as stated in [7], 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 [14, 40], rely on the Fiat–Shamir transform and the forking lemma heavily. First, an interactive protocol is proposed and its security and (a variant of) 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 this loss. We note that the security loss may also apply when knowledge soundness is proven. That is the case for Sonic, whose security proof relies on so-called witness-extended emulation. Authors of Plonk worked around this problem by showing their protocol security 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 $\mathbf{P}_{\mathsf{FS}}$ and $\mathbf{S}_{\mathsf{FS}}$. The proof is inspired by simulation-extractability and simulation-soundness proofs from [22], with major modifications, which were required as [22] considers only Σ-protocols that are undoubtedly simpler protocols than the considered proof systems.

Generalising Boneh-Boyen-Goh [10] uber assumption. To show that Plonk is zero-knowledge we rely on a variant of BBG's *uber assumption*. In its original version, the assumption assures that some polynomial evaluation (on a random, unknown point) represented in a bilinear pairing target group $\mathbb{G}_T$ is indistinguishable from a random element. In our variant, we modify two things.

- Firstly, the polynomial evaluation can be represented in other groups than $\mathbb{G}_T$; here we use $\mathbb{G}_1$;
- Secondly, the distinguisher is given not a single polynomial evaluation, but a number of polynomials. That is, it either gets k polynomial evaluations representations or k random numbers.

We show security of the generalized uber assumption directly in the generic group model.

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 and Section 6 we show that Plonk and Sonic fulfil requirements of our framework and in fact is simulation extractable.1

1.4 Related Work

There are many results on simulation extractability for non-interactive zero-knowledge proofs (NIZKs). First, Groth [30] noticed that a (black-box) simulation extractable NIZK is universally-composable (UC) [15]. Then Dodis et al. [21] introduced a notion of (black-box) *true simulation extractability* and showed that no NIZK can be UC-secure if it does not satisfy this property. In the context of zkSNARKs it is important to mention such works as the first simulation-extractable zkSNARK by Groth and Maller [34] and SE zkSNARK for QAP by Lipmaa [39]. Kosba's et al. [36] 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. [2]

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 [32], by introducing minor modifications and using stronger assumptions [3,12]. Interestingly, although such modifications hurt performance of the proof system, the resulting zkSNARKs are still more efficient than the first SE zkSNARK [34], see [3]. Recently, [5] showed that the original Groth's proof system from [32] is weakly SE and randomisable.

Forking lemma generalizations. In [11] Bootle et al. proposed a novel inner-product argument which security relies on, so-called, witness-extended emulation. To show that property, the authors proposed a new version of forking lemma, which gives a lower bound on probability that a tree finding algorithm is able to produce a tree of acceptable transcripts by rewinding a conversation between a (potentially malicious) prover and verifier.

Although the result of presented in that paper is dubbed "forking lemma" it differs from forking lemmas known from e.g. [7,42]. First of all, the forking lemmas in these papers analyse probability of building a tree of acceptable transcripts for Fiat–Shamir based non-interactive proof systems, while the protocol presented by Bootle et al. is intended to work for interactive proof systems.

Importantly, it is not obvious how the result of Bootle et al. can be used to show security of non-interactive protocols as it relies on interactive provers whose proving strategies are more limited than proving strategies of non-interactive ones. For example, if a challenge given by the verifier does not suit an interactive prover, it can only try to finish a proof with it or abort. On the other hand, a non-interactive prover has far wider scope of possible actions—when the protocol is non-interactive the prover may adapt its strategy respectively to the random oracle outputs. For example, seeing a response h after sending k-th message it may decide to "go back" a number of steps, e.g. prior sending k'-th message ($k' \leq k$), provide different messages hoping for better suited response h' on its k-th message. Furthermore, the adversary may even interrupt the proof and try again with another instance. All of these actions would make the proof unacceptable for an interactive verifier, however they may give a perfectly fine proof for a non-interactive one.

2 Preliminaries

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

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

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

Denote by $\mathcal{R} = \{\mathbf{R}\}$ a family of relations. We assume that if $\mathbf{R}$ comes with any auxiliary input, it is benign. Directly from the description of $\mathbf{R}$ one learns security parameter λ and other necessary information like public parameters p containing description of a group $\mathbb{G}$, if the relation is a relation of group elements (as it usually is in case of zkSNARKs).

Bilinear groups. A bilinear group generator $\mathsf{Pgen}(1^\lambda)$ returns public parameters $\mathsf{p} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2)$, where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are additive cyclic groups of prime order $p = 2^{\Omega(\lambda)}$, $[1]_1, [1]_2$ are generators of $\mathbb{G}_1$, $\mathbb{G}_2$, resp., and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate PPT-computable bilinear pairing. We assume the bilinear pairing to be Type-3, i.e., that there is no efficient isomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$ or from $\mathbb{G}_2$ to $\mathbb{G}_1$. We use the by now standard bracket notation, i.e., we write $[a]_\iota$ to denote ag_ι where g_ι is a fixed generator of $\mathbb{G}_\iota$. We denote $\hat{e}([a]_1, [b]_2)$ as $[a]_1 \bullet [b]_2$. Thus, $[a]_1 \bullet [b]_2 = [ab]_T$. We freely use the bracket notation with matrices, e.g., if $\boldsymbol{AB} = \boldsymbol{C}$ then $\boldsymbol{A}[\boldsymbol{B}]_\iota = [\boldsymbol{C}]_\iota$ and $[\boldsymbol{A}]_1 \bullet [\boldsymbol{B}]_2 = [\boldsymbol{C}]_T$. Since every algorithm $\mathcal{A}$ takes as input the public parameters we skip them when describing $\mathcal{A}$'s input. Similarly, we do not explicitly state that each protocol starts with generating these parameters by Pgen.

2.1 Computational assumptions.

Discrete-log assumptions. Security of Plonk and Sonic relies on two discrete-log based security assumptions—(q_1, q_2)-dlog assumption and its extended version with negative exponents (q_1, q_2)-ldlog assumption[4].

Definition 1 ((q_1, q_2)-dlog assumption). *Let $\mathcal{A}$ be a* PPT *adversary that gets as input $[1, \chi, \ldots, \chi^{q_1}]_1, [1, \chi, \ldots, \chi^{q_2}]_2$, for some randomly picked $\chi \in \mathbb{F}_p$, 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 2 ((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*

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

BBG uber assumption. Also, to be able to show computational honest verifier zero knowledge of Plonk in the standard model, what is required by our reduction, we rely on the *uber assumption* introduced by Boneh et al. [10] as presented by Boyen in [13].

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

For a function f and vector $(x_1, \ldots, x_c)$ we write $f(\mathsf{R})$ to denote application of f to each element of R, i.e.

$$f(\mathsf{R}) = (f(\mathsf{r}_1(x_1, \ldots, x_c), \ldots, f(\mathsf{r}_r(x_1, \ldots, x_c)).$$

Similarly for applying f to S and T.

Definition 3 (Independence of $\mathsf{R}, \mathsf{S}, \mathsf{T}$). *Let $\mathsf{R}, \mathsf{S}, \mathsf{T}$ be defined as above. We say that polynomial $\mathsf{f} \in \mathbb{F}_p[X_1, \ldots, X_c]$ is* dependent *on $\mathsf{R}, \mathsf{S}, \mathsf{T}$ if there exists $rs + t$ constants $a_{i,j}, b_k$ such that $\mathsf{f} = \sum_{i=1}^{r} \sum_{j=1}^{s} a_{i,j} \mathsf{r}_i \mathsf{s}_j + \sum_{k=1}^{t} b_k \mathsf{t}_k$. We say that f is* independent *if it is not dependent.*

To show (standard-model) zero knowledge of Plonk we utilize a generalization of Boneh-Boyen-Goh's *uber assumption* [10] stated as follows (the changed element has been put into a dashbox)

Definition 4 ($(\mathsf{R}, \mathsf{S}, \mathsf{T}, \mathsf{F}, 1)$-uber assumption). *Let $\mathsf{R}, \mathsf{S}, \mathsf{T}$ be defined as above, $(x_1, \ldots, x_c, y_1, \ldots, y_d) \leftarrow_\$ \mathbb{F}_p^{c+d}$ and let F be a cardinality-d set of pair-wise independent polynomials which are also independent of $(\mathsf{R}, \mathsf{S}, \mathsf{T})$, cf. Definition 3. Then, for any* PPT *adversary $\mathcal{A}$*

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

Compared to the original uber assumptions, there are two major changes. First, we require not target group $\mathbb{G}_T$ elements to be indistinguishable, but elements of $\mathbb{G}_1$. Second, Boneh et al.'s assumption works for distinguishers who are given only one challenge polynomial f, i.e. $|\mathsf{F}| = 1$. We prove our variant of the uber assumption in the generic group model, see Theorem 3.

Proofs by Game-Hoping. Proofs by *game hoping* is a method of writing proofs popularised by e.g. Shoup [46] and Dent [20]. The method relies on the following lemma.

Lemma 1 (Difference lemma, [46, Lemma 1]). *Let $\mathsf{A}, \mathsf{B}, \mathsf{F}$ be events defined in some probability space, and suppose that $\mathsf{A} \wedge \overline{\mathsf{F}} \iff \mathsf{B} \wedge \overline{\mathsf{F}}$. Then $|\Pr[\mathsf{A}] - \Pr[\mathsf{B}]| \leq \Pr[\mathsf{F}]$.*

[4] Note that [40] dubs their assumption *a dlog assumption*. We changed that name to distinguish it from the more standard dlog assumption used in [27]. "l" in *ldlog* relates to use of Laurent polynomials in the assumption.

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

$\chi \leftarrow\!\!\$\ \mathbb{F}_p^2$
return $\left[1, \ldots, \chi^{n+2}\right]_1, [\chi]_2$

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

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

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

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

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

$\boldsymbol{r} \leftarrow \mathbb{F}_p^{|\boldsymbol{z}|}$
for $i \in [1 \,..\, |\boldsymbol{z}|]$ **do**
$\quad$ **if** $\sum_{i=1}^{|\boldsymbol{z}|} r_i \cdot \left[\sum_{j=1}^{t_j} \gamma_i^{j-1} c_{i,j} - \sum j = 1^{t_j} s_{i,j}\right]_1 \bullet [1]_2 +$
$\quad \sum_{i=1}^{|\boldsymbol{z}|} r_i z_i o_i \bullet [1]_2 \neq \left[-\sum_{i=1}^{|\boldsymbol{z}|} r_i o_i\right]_1 \bullet [\chi]_2$ **then**
$\quad\quad$ **return** 0
return 1.

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

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

$\alpha, \chi \leftarrow\!\!\$\ \mathbb{F}_p^2$
return $\left[\{\chi^i\}_{i=-\mathsf{n}}^{\mathsf{n}}, \{\alpha\chi^i\}_{i=-\mathsf{n}, i\neq 0}^{\mathsf{n}}\right]_1,$
$\quad \left[\{\chi^i, \alpha\chi^i\}_{i=-\mathsf{n}}^{\mathsf{n}}\right]_2, [\alpha]_T$

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

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

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

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

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

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

Fig. 2: $\mathbf{PC_S}$ polynomial commitment scheme.

2.2 Algebraic Group Model

The algebraic group model (AGM) introduced in [25] 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*.

2.3 Polynomial commitment.

In the polynomial commitment scheme $\mathbf{PC} = (\mathsf{KGen}, \mathsf{Com}, \mathsf{Op}, \mathsf{Vf})$ the committer C can convince the receiver R that some polynomial f which C committed to evaluates to s at some point z chosen by R. Plonk and Sonic use variants of the KZG polynomial commitment scheme [35]. We denote the first by $\mathbf{PC_P}$, presented in Fig. 1, and the latter by $\mathbf{PC_S}$, presented in Fig. 2. The key generation algorithm KGen takes as input a security parameter 1^λ and a parameter max which determines the maximal degree of the committed polynomial. We assume that max can be read from the output SRS.

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

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

and $\boldsymbol{o}, \boldsymbol{o}' \in \mathbb{F}_p^l$ be the commitment openings. Then for every PPT adversary $\mathcal{A}$

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

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

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

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

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 7 respectively.

Commitment of knowledge For every PPT adversary $\mathcal{A}$ who produces commitment c, evaluation point z, evaluation s and opening o there exists a PPT extractor Ext such that

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

In that case we say that **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. [40] shows, using AGM, that $\mathbf{PC_S}$ is a commitment of knowledge. The same reasoning could be used to show that property for $\mathbf{PC_P}$. We skip a proof for that fact due to the lack of space.

2.4 Zero knowledge

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

More precisely, let $\mathcal{R}(1^\lambda) = \{\mathbf{R}\}$ be a family of NP relations. Denote by $\mathcal{L}_{\mathbf{R}}$ the language determined by $\mathbf{R}$. Let P and V be PPT algorithms, the former called *prover* and the latter *verifier*. We allow our proof system to have a setup, i.e. there is a KGen algorithm that takes as input the relation description $\mathbf{R}$ and outputs a common reference string srs. We denote by $\langle \mathsf{P}(\mathbf{R}, \mathsf{srs}, \mathsf{x}, \mathsf{w}), \mathsf{V}(\mathbf{R}, \mathsf{srs}, \mathsf{x}) \rangle$ a *transcript* (also called *proof*) π of a conversation between P with input $(\mathbf{R}, \mathsf{srs}, \mathsf{x}, \mathsf{w})$ and V with input $(\mathbf{R}, \mathsf{srs}, \mathsf{x})$. We write $\langle \mathsf{P}(\mathbf{R}, \mathsf{srs}, \mathsf{x}, \mathsf{w}), \mathsf{V}(\mathbf{R}, \mathsf{srs}, \mathsf{x}) \rangle = 1$ if in the end of the transcript the verifier V returns 1 and say that V accepts it. We sometimes abuse notation and write $\mathsf{V}(\mathbf{R}, \mathsf{srs}, \mathsf{x}, \pi) = 1$ to denote a fact that π is accepted by the verifier. (This is especially handy when the proof system is non-interactive, i.e. the whole conversation between the prover and verifier consists of a single message π sent by P).

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

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

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

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

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

Sometimes a stronger notion of soundness is required—except requiring that the verifier rejects proofs of statements outside the language, we request from the prover to know a witness corresponding to the proven statement. This property is formalised by the so-called *knowledge soundness*. That is, we call an interactive proof system $\mathbf{\Psi}$ *knowledge-sound* if for any $\mathbf{R} \in \mathcal{R}(1^\lambda)$ and a PPT adversary $\mathcal{A}$

$$\Pr\left[\begin{matrix}\mathsf{V}(\mathbf{R}, \mathsf{srs}, \mathsf{x}, \pi) = 1, \\ \mathbf{R}(\mathsf{x}, \mathsf{w}) = 0\end{matrix} \middle| \begin{matrix}\mathsf{srs} \leftarrow \mathsf{KGen}(\mathbf{R}), \mathsf{x} \leftarrow \mathcal{A}(\mathbf{R}, \mathsf{srs}), \\ (\mathsf{w}, \pi) \leftarrow \mathsf{Ext}^{\langle \mathcal{A}(\mathbf{R}, \mathsf{srs}, \mathsf{x}), \mathsf{V}(\mathbf{R}, \mathsf{srs}, \mathsf{x})\rangle}(\mathbf{R}, \mathsf{x})\end{matrix}\right] \le \mathsf{negl}(\lambda)\ ,$$

Zero knowledge. We call a proof system $\mathbf{\Psi}$ *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

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

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

An SRS srs comes with a secret string called *trapdoor* td that allows the simulator to produce a simulated proof. In that case algorithm $\mathsf{KGen}(\mathbf{R})$ outputs $(\mathsf{srs}, \mathsf{td})$ and td is given to the simulator. In this paper we distinguish simulators that requires a trapdoor to simulate and those that do not. We call the former *SRS-simulators* and denote them by $\mathsf{Sim}_{\mathsf{td}}$.

Occasionally, a weaker version of zero knowledge is sufficient. So called *honest verifier zero knowledge* (HVZK) 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 and zero-knowledge in the random oracle model by using the Fiat–Shamir transformation.

Idealised verifier and verification equations Usually the verifier V verifies messages sent by the prover by checking a number of equations that depend on the instance, SRS and the proof. We call these equations *verification equations* and denote by ve_i, for i being an index of the equation. We require that an acceptable proof yields $\mathsf{ve}_i = 0$.

In the proof systems we consider—Plonk and Sonic—verification equations can be interpreted as group representations of polynomial evaluations at the trapdoor χ. In other words, the verifier checks that $\mathsf{ve}_i(\chi) = 0$. Gabizon et al. [27] formalized a notion of an *idealised verifier* who, instead of checking that polynomial $\mathsf{ve}_i(X)$ evaluates to 0 at χ, just checks that $\mathsf{ve}_i(X)$ is a zero polynomial.

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

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

Another property that sigma protocols may have is a unique response property [23] 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

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

If this property holds even against unbounded adversaries, it is called *strict*, cf. [22]. Later on we call protocols that follows this notion *ur-protocols*. For the sake of completeness we note that many sigma protocols, like e.g. Schnorr's protocol [44], fulfil this property.

2.5 From interactive to non-interactive—Fiat–Shamir transformation

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

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

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

2.6 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 so-called *simulation extractability* which assures that no adversary can produce a valid proof without knowing the corresponding witness. This must hold even if the adversary is allowed to see polynomially many simulated proofs for any statements it wishes.

Definition 5 ((Weak) simulation-extractable NIZK, [22]). *Let $\mathbf{\Psi} = (\mathsf{KGen}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ be a computationally special-sound HVZK proof and $\mathbf{\Psi}_{\mathsf{FS}} = (\mathsf{KGen}_{\mathsf{FS}}, \mathsf{P}_{\mathsf{FS}}, \mathsf{V}_{\mathsf{FS}}, \mathsf{Sim}_{\mathsf{FS}})$ be $\mathbf{\Psi}$ transformed by the Fiat–Shamir transform. We say that $\mathbf{\Psi}_{\mathsf{FS}}$ 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 $\mathbf{\Psi}$ with probability* acc*, that is*

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

probability

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

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 $\mathcal{A}$'s queries to $\mathcal{H}$ and $\mathcal{H}$'s answers.*

Consider a sigma protocol $\Sigma = (\mathsf{P}, \mathsf{V}, \mathsf{Sim})$ that is specially sound and has a unique response property. Let $\Sigma_{\mathsf{FS}} = (\mathsf{P}_{\mathsf{FS}}, \mathsf{V}_{\mathsf{FS}}, \mathsf{Sim}_{\mathsf{FS}})$ be a NIZK obtained by applying the Fiat–Shamir transform to Σ. Faust et al. [22] show that every such Σ_{FS} is simulation-extractable. This result is presented in Appendix A along with the instrumental forking lemma, cf. [7].

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

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

Fig. 3: Generalised forking algorithm $\mathsf{GF}^m_{\mathcal{Z}}$

3 Towards simulation extractability of 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 transcripts for the extractor to work and are not special sound. In this part we generalize the forking lemma, special soundness, and the unique response property to make them compatible with multi-message protocols.

3.1 Generalised forking lemma.

First of all, although dubbed "general", Lemma 11 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 $\mathbf{P}$ we need to obtain at least $\mathsf{n} + 3$ valid proofs, and even more for $\mathbf{S}$. Here we propose a generalisation of the general forking lemma that given probability of producing an accepting transcript acc lower-bounds the probability of generating a *tree of accepting transcripts* T, which allows to extract a witness.

Definition 6 (Tree of accepting transcripts, cf. [11]). *Consider a $(2\mu + 1)$-message interactive proof system $\boldsymbol{\Psi}$. An $(n_1, \ldots, n_\mu)$-tree of accepting transcript is a tree where each node on depth i, for $i \in [1\,..\,\mu + 1]$, is an i-th prover's message in an acceptable transcript; edges between the nodes are labeled with verifier's challenges, such that no two edges on the same depth have the same label; and each node on depth i has $n_i - 1$ siblings and n_{i+1} children. Altogether, the tree consists of $N = \prod_{i=1}^{\mu} n_i$ branches, which makes N acceptable transcripts. We require $N = \mathsf{poly}(\lambda)$.*

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

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

Let $\mathsf{GF}^m_{\mathcal{Z}}$ denote the algorithm described in Fig. 3 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

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

$$\mathsf{frk}(y) = \Pr\left[b = 1 \mid (b, \boldsymbol{s}) \leftarrow \mathsf{GF}^m_{\mathcal{Z}}(y, h_1, \ldots, h_q)\right] .$$

We start by claiming that for all y

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

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

$$\begin{aligned}
\mathsf{frk} = \mathbb{E}\left[\mathsf{frk}(y)\right] &\geq \mathbb{E}\left[\frac{\mathsf{acc}(y)^m}{q^{m-1}} - \mathsf{acc}(y) \cdot \left(1 - \frac{h!}{(h-m)! \cdot h^m}\right)\right] && (2)\\
&\geq \frac{\mathbb{E}\left[\mathsf{acc}(y)\right]^m}{q^{m-1}} - \mathbb{E}\left[\mathsf{acc}(y)\right] \cdot \left(1 - \frac{h!}{(h-m)! \cdot h^m}\right) && (3)\\
&= \frac{\mathsf{acc}^m}{q^{m-1}} - \mathsf{acc} \cdot \left(1 - \frac{h!}{(h-m)! \cdot h^m}\right) . && (4)
\end{aligned}$$

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

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

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

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

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

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

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

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

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

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

By Lemma 12 we get

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

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

We now use the Hölder inequality, cf. Lemma 13, for $x_i = \mathbb{E}[X_i]$, $y_i = 1$, $p = m$, 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

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

□

To highlight importance of the generalised forking lemma we describe how we use it in our simulation-extractability proof. Let $\boldsymbol{\Psi}$ be a computationally special sound proof system where for an instance x the corresponding witness can be extracted from an $(1, \ldots, 1, n_k, 1, \ldots, 1)$-tree of accepting transcripts. Let $\mathcal{A}$ be the simulation-extractability adversary that outputs an acceptable proof with probability at least acc. (Although we use the same acc to denote probability of $\mathcal{Z}$ outputing a non-zero i and probability of $\mathcal{A}$ outputing an acceptable proof we claim that these probabilities are exactly the same what comes from how we define $\mathcal{Z}$.) Let $\mathcal{A}$ produce an acceptable proof $\pi_{\mathcal{A}}$ for instance $\mathsf{x}_{\mathcal{A}}$; r be $\mathcal{A}$'s randomness; Q the list of queries submitted by $\mathcal{A}$ along with simulator's Sim_π answers; and $Q_{\mathcal{H}}$ be the list of all random oracle queries made by $\mathcal{A}$. All of these are given to the extractor Ext that internally runs the forking algorithm $\mathsf{GF}_{\mathcal{Z}}^{n_k}$. Algorithm $\mathcal{Z}$ takes $(\mathbf{R}, \mathsf{srs}, \mathcal{A}, Q, r)$ as input y and $Q_{\mathcal{H}}$ as input $h_1^1, \ldots, h_q^1$. (For the sake of completeness, we allow $\mathsf{GF}_{\mathcal{Z}}^{n_k}$ to pick $h_{l+1}^1, \ldots, h_q^1$ responses if $Q_{\mathcal{H}}$ has only $l < q$ elements.)

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

We note that in the original forking lemma the forking algorithm F, cf. Fig. 4, gets only as input y and elements $h_1^1, \ldots, h_q^1$ 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 the change only notational.

We also note that the general forking lemma proposed in Lemma 2 works for protocols which have extractable witness from a $(1, \ldots, 1, n_k, 1, \ldots, 1)$-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.

3.2 Unique-response protocols.

Another problem comes with another assumption required by Faust et al.—the unique response property of the transformed sigma protocol. The original Fischlin's formulation, although suitable for applications presented in [22, 23], 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 the prover needs to answer differently on different verifier's challenges, as otherwise the protocol could have fewer rounds. Another problem arises 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 SRS what also complicates things considerably.

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

Definition 7 (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 the Fiat–Shamir transform, Denote by $a_1, \ldots, a_\mu, a_{\mu+1}$ protocol messages output by the prover, We say that $\boldsymbol{\Psi}$ has* unique responses from i on *if for any* PPT *adversary $\mathcal{A}$:*

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

Intuitively, a protocol is i-ur if it is infeasible for a PPT adversary to produce a pair of acceptable and different proofs π, π' that are the same on first i messages. We note that the definition above is independent on whether the proof system $\boldsymbol{\Psi}$ utilises SRS (and compounds of the SRS-generating KGen algorithm) or not.

3.3 Computational special soundness

Note that the special soundness property (as usually defined) holds for all—even computationally unbounded—adversaries. Unfortunately, since a simulation trapdoors for $\mathbf{P}$ and $\mathbf{S}$ exist, the protocols cannot be special sound in that regard. This comes since 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 *computational special 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 8 (Computational special 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* $(\varepsilon_{\mathsf{ss}}, (n_1, \ldots, n_\mu))$-computationally special sound *if there exists an extractor $\mathsf{Ext}_{\mathsf{tree}}$ that given an $(n_1, \ldots, n_\mu)$-tree of acceptable transcripts T and instance x output by some* PPT *adversary $\mathcal{A}(\mathbf{R}, \mathsf{srs})$, for $\mathsf{srs} \leftarrow\!\!\$\, \mathsf{KGen}(\mathbf{R})$, outputs w such that $\mathbf{R}(\mathsf{x}, \mathsf{w}) = 1$ with probability at least $1 - \varepsilon_{\mathsf{ss}}$.*

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

Equipped with definitional framework of Section 3 we are ready to present the main result of this paper—simulation extractability of multi-round protocols.

Theorem 1 (Simulation-extractable multi-message protocols). *Let $\boldsymbol{\Psi} = (\mathsf{KGen}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ be an interactive $(2\mu+1)$-message proof system for $\mathcal{R}(1^\lambda)$ that is honest verifier zero-knowledge in the standard model[5], has k-ur property with security $\varepsilon_{\mathsf{ur}}$, is $(n_1, \ldots, n_\mu)$-special sound for $n_i = 1, i \in [1\,..\,\mu] \setminus \{k\}$ and $n_k = n$.*

Let $\mathcal{H}\colon \{0,1\}^* \to \{0,1\}^\lambda$ be a random oracle. Then $\boldsymbol{\Psi}_{\mathsf{FS}}$ 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*

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

for some negligible ε.

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

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

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

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

Game G_0: This is a simulation extraction game played between an adversary $\mathcal{A}$ who has given access to a random oracle $\mathcal{H}$ and simulator $\mathbf{\Psi}_{\mathsf{FS}}.\mathsf{Sim}$. There is also an extractor Ext that, from a proof $\pi_{\mathcal{A}}$ for instance $\mathsf{x}_{\mathcal{A}}$ output by the adversary and from transcripts of $\mathcal{A}$'s operations is tasked to extract a witness $\mathsf{w}_{\mathcal{A}}$ such that $\mathbf{R}(\mathsf{x}_{\mathcal{A}}, \mathsf{w}_{\mathcal{A}})$ holds. $\mathcal{A}$ wins if it manages to produce an acceptable proof and the extractor fails to reveal the corresponding witness. In the following game hops we upper-bound the probability that this happens.

Game G_1: This is identical to G_0 except that now the game is aborted if there is a simulated proof π_{Sim} for $\mathsf{x}_{\mathcal{A}}$ such that $(\mathsf{x}_{\mathcal{A}}, \pi_{\mathsf{Sim}}[1..k]) = (\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}[1..k])$. That is, the adversary in its final proof reuses 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, $\Pr\left[\mathsf{G}_0 \wedge \overline{\mathsf{Err}_{\mathsf{ur}}}\right] = \Pr\left[\mathsf{G}_1 \wedge \overline{\mathsf{Err}_{\mathsf{ur}}}\right]$ and, from the difference lemma, cf. Lemma 1,

$$|\Pr[\mathsf{G}_0] - \Pr[\mathsf{G}_1]| \leq \Pr[\mathsf{Err}_{\mathsf{ur}}] \ .$$

Thus, to show that the transition from one game to another introduces only minor change in probability of $\mathcal{A}$ winning it should be shown that $\Pr[\mathsf{Err}_{\mathsf{ur}}]$ is small.

We can assume that $\mathcal{A}$ queried the simulator on the instance it wishes to output—$\mathsf{x}_{\mathcal{A}}$. We show a reduction $\mathcal{R}_{\mathsf{ur}}$ that utilises $\mathcal{A}$, who outputs a valid proof for $\mathsf{x}_{\mathcal{A}}$, to break the k-ur property of $\mathbf{\Psi}$. Let $\mathcal{R}_{\mathsf{ur}}$ run $\mathcal{A}$ 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 $\mathcal{A}$ makes a fake proof $\pi_{\mathcal{A}}$ for $\mathsf{x}_{\mathcal{A}}$, $\mathcal{R}_{\mathsf{ur}}$ looks through lists Q and $Q_{\mathcal{H}}$ until it finds $\pi_{\mathsf{Sim}}[0..k]$ such that $\pi_{\mathcal{A}}[0..k] = \pi_{\mathsf{Sim}}[0..k]$ and a random oracle query $\pi_{\mathsf{Sim}}[k].\mathsf{ch}$ on $\pi_{\mathsf{Sim}}[0..k]$.
- $\mathcal{R}_{\mathsf{ur}}$ returns two proofs for $\mathsf{x}_{\mathcal{A}}$:

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

If $\pi_1 = \pi_2$, then $\mathcal{A}$ fails to break simulation 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 $(1, \ldots, 1, n, 1, \ldots, 1)$-tree of accepting transcripts T by rewinding $\mathcal{A}$. Denote that event by $\mathsf{Err}_{\mathsf{frk}}$.

$\mathsf{G}_1 \mapsto \mathsf{G}_2$: Note that for every acceptable proof $\pi_{\mathcal{A}}$, we may assume that whenever $\mathcal{A}$ outputs in Round k message $\pi_{\mathcal{A}}[k]$, then $(\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}[1..k])$ random oracle query that was made by the adversary, not the simulator[6], i.e. there is no simulated proof π_{Sim} on $\mathsf{x}_{\mathsf{Sim}}$ such that $(\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}[1..k]) = (\mathsf{x}_{\mathsf{Sim}}, \pi_{\mathsf{Sim}}[1..k])$. Otherwise, the game would be already interrupted by the error event in Game G_1. As previously,

$$|\Pr[\mathsf{G}_1] - \Pr[\mathsf{G}_2]| \leq \Pr[\mathsf{Err}_{\mathsf{frk}}] \ .$$

[6] [22] calls these queries *fresh*.

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

To assure that in the first execution of $\mathcal{Z}$ the adversary $\mathcal{A}$ produce the same $(\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}})$ as in the extraction game, $\mathcal{Z}$ provides $\mathcal{A}$ with the same randomness r and answers queries to the random oracle and simulator with responses pre-recorded 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 i of a random oracle query that was used by $\mathcal{A}$ to compute the challenge $\pi[k].\mathsf{ch} = \mathcal{H}(\pi_{\mathcal{A}}[0..k])$ it had to answer in the $(k+1)$-th round and adversary's transcript, denoted by s_1 in GF's description. If no such query took place $\mathcal{Z}$ outputs $i = 0$.

Then new random oracle responses are picked for queries indexed by $i, \ldots, q$ and the adversary is rewound to the point just prior it gets the response to RO query $\pi_{\mathcal{A}}[0..k]$. The adversary gets a random oracle response from a new set of responses $h_i^2, \ldots, h_q^2$. If the adversary requests a simulated proof after seeing h_i^2 then $\mathcal{Z}$ computes the simulated proof on its own. Eventually, $\mathcal{Z}$ outputs index i' of a query that was used by the adversary to compute $\mathcal{H}(\pi_{\mathcal{A}}[0..k])$, and a new transcript s_2. $\mathcal{Z}$ is run $n+1$ times with different random oracle responses. If a tree T of $n+1$ transcripts is build then Ext runs internally the tree extractor $\mathsf{Ext}_{\mathsf{tree}}(\mathsf{T})$ and outputs what it returns.

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

Denote by $\widetilde{\mathsf{acc}}$ the probability that $\mathcal{A}$ outputs a proof that is accepted and does not break k-ur-ness of $\mathbf{\Psi}$. Denote by $\widetilde{\mathsf{acc}}'$ probability that algorithm $\mathcal{Z}$, defined in the lemma, produces an accepting proof with a fresh challenge after Round k. Given the discussion above, we can state that $\widetilde{\mathsf{acc}} = \widetilde{\mathsf{acc}}'$.

Next, from the generalised forking lemma, cf. Lemma 2,

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

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

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

Game G_3: This is identical to G_2 except that now the game is aborted if $\mathsf{Ext}_{\mathsf{tree}}(\mathsf{T})$ run by Ext fails to extract the witness. 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 $\mathbf{\Psi}$ is special-sound the probability of $\mathsf{Err}_{\mathsf{ss}}$ is upper-bounded by some negligible $\varepsilon_{\mathsf{ss}}$.

Game G_3 is aborted when it is impossible to extract the correct witness from T, hence the adversary $\mathcal{A}$ cannot win. Thus, by the game-hoping argument,

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

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

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

Since $\widetilde{\mathsf{acc}}$ is probability of $\mathcal{A}$ outputting acceptable transcript that does not break k-ur-ness of $\mathbf{\Psi}$, then $\widetilde{\mathsf{acc}} \geq \mathsf{acc} - \varepsilon_{\mathsf{ur}}$, where acc is the probability of $\mathcal{A}$ outputing an acceptable proof as defined in Definition 5.

It thus holds

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

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

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

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 2.3 and Lemma 3. This is then used to show that $\mathbf{P}$ has the 3-ur property, cf. Lemma 4.

Next, we show that $\mathbf{P}$ is computationally 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. This result is shown in the AGM, cf. Lemma 5.

Given special-soundness of $\mathbf{P}$, we use the fact that it is also 3-ur and show, in a similar fashion to [22], that it is simulation-extractable. That is, we build reductions that given a simulation extractability adversary $\mathcal{A}$ either break the protocol's unique response property or break the dlog assumption, if extracting a valid witness from a tree of transcripts is impossible. See Corollary 1.

5.1 Plonk protocol rolled out

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

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

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

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

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

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

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

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

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

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

Coefficients k_1, k_2 are such that $H, k_1 \cdot H, k_2 \cdot H$ are different cosets of $\mathbb{F}^*$, thus they define $3 \cdot \mathsf{n}$ different elements. [27] notes that it is enough to set k_1 to a quadratic residue and k_2 to a quadratic non-residue.

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

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

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

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

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

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

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

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

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

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

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

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

$$\mathsf{t}(X) = \\ (\mathsf{a}(X)\mathsf{b}(X)\mathsf{q}_\mathsf{M}(X) + \mathsf{a}(X)\mathsf{q}_\mathsf{L}(X) + \mathsf{b}(X)\mathsf{q}_\mathsf{R}(X) + \mathsf{c}(X)\mathsf{q}_\mathsf{O}(X) + \mathsf{PI}(X) + \mathsf{q}_\mathsf{C}(X))\frac{1}{\mathsf{Z}_\mathsf{H}(X)} + \\ + ((\mathsf{a}(X) + \beta X + \gamma)(\mathsf{b}(X) + \beta k_1 X + \gamma)(\mathsf{c}(X) + \beta k_2 X + \gamma)\mathsf{z}(X))\frac{\alpha}{\mathsf{Z}_\mathsf{H}(X)}$$

$$- (\mathsf{a}(X) + \beta\mathsf{S}_{\sigma 1}(X) + \gamma)(\mathsf{b}(X) + \beta\mathsf{S}_{\sigma 2}(X) + \gamma)(\mathsf{c}(X) + \beta\mathsf{S}_{\sigma 3}(X) + \gamma)\mathsf{z}(X\omega))\frac{\alpha}{\mathsf{Z}_{\mathsf{H}}(X)}$$
$$+ (\mathsf{z}(X) - 1)\mathsf{L}_1(X)\frac{\alpha^2}{\mathsf{Z}_{\mathsf{H}}(X)}$$

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

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

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

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

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

Compute the linearisation polynomial

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

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

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

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

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

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

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

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

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

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

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 $[E]_1$

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

Step 12 Check whether the verification equation holds

$$([\mathsf{W}_{\mathfrak{z}}(\chi)]_1 + u\cdot[\mathsf{W}_{\mathfrak{z}\omega}(\chi)]_1)\bullet[\chi]_2 - (\mathfrak{z}\cdot[\mathsf{W}_{\mathfrak{z}}(\chi)]_1 + u\mathfrak{z}\omega\cdot[\mathsf{W}_{\mathfrak{z}\omega}(\chi)]_1 + [F]_1 - [E]_1)\bullet[1]_2 \;=\; 0\,. \quad (7)$$

The verification equation is a batched version of the verification equation from [35] 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).

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

Lemma 3. *Let* $\mathsf{PC_P}$ *be a batched version of a KZG polynomial commitment [35] as described in [27] 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

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

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 [27, 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 SRS, cf. [27, Lemma 2.2] for more details.

For polynomials $\mathbf{f} = \mathsf{f}_1, \ldots, \mathsf{f}_k$, evaluation point z, evaluation result $\boldsymbol{s} = s_1, \ldots, s_k$, random γ, and opening $\mathsf{o}(X)$ the idealised check verifies that

$$\sum_{i=1}^{k} \gamma^{i-1} \mathsf{f}_i(X) - \sum_{i=1}^{k} \gamma^{i-1} s_i \equiv \mathsf{o}(X)(X - z)\,. \tag{8}$$

Since $\mathsf{o}(X)(X - z) \in \mathbb{F}_p[X]$ then from the uniqueness of polynomial composition, there is only one $\mathsf{o}(X)$ 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:

$$\sum_{i=1}^{k} \gamma^{i-1} \cdot \mathsf{f}_i(X) - \sum_{i=1}^{k} \gamma^{i-1} \cdot s_i + r' \left(\sum_{i=1}^{k'} \gamma'^{i-1} \cdot \mathsf{f'}_i(X) - \sum_{i=1}^{k'} \gamma'^{i-1} \cdot s'_i \right) \equiv \mathsf{o}(X)(X - z) + r'\mathsf{o}'(X)(X - z') \tag{9}$$

Since r' has been picked at random, probability that Eq. (9) holds while either

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

or

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

is negligible [27]. This brings the proof back to Step 1 above. □

5.3 Unique response property

Lemma 4. *Let* $\mathbf{PC_P}$ *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* 3-ur *property is at most* $\varepsilon_{\mathsf{k}} + 2 \cdot \varepsilon_{\mathsf{bind}} + \varepsilon_{\mathsf{op}}$.

Proof (sketch). Let $\mathcal{A}(\mathbf{R}, \mathsf{srs} = ([1, \chi, \ldots, \chi^{\mathsf{n}+2}]_1, [\chi]_2))$ be an adversary tasked to break the 3-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 the AGM and arguments similar to [40], we argue that $\mathbf{PC_P}$ is a commitment of knowledge. That is, an AGM adversary $\mathcal{A}$ that outputs a commitment c' which it can later open, knows a polynomial f of degree-($\leq \mathsf{n} + 2$) 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 no more than $\mathsf{n} + 2$ overlapping points between $\mathsf{f}(X)$ and $\mathsf{t}(X)$, 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 $\mathbf{PC_P}.\mathsf{Vf}$ then the evaluation binding property of $\mathbf{PC_P}$ 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 $\mathbf{PC}_\mathbf{P}$ 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

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

□

5.4 Computational special soundness

Lemma 5. P *is* $(\varepsilon_{\mathsf{ss}}, (1, 1, \mathsf{n}+3, 1))$*-computational special sound for*

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

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

Proof. Let srs be $\mathbf{P}$'s SRS and denote by srs_1 all SRS's $\mathbb{G}_1$-elements; that is, $\mathsf{srs}_1 = [1, \chi, \ldots, \chi^{\mathsf{n}+2}]_1$. Let $\mathcal{A}$ be an algebraic adversary that for produces a statement x and $(1, 1, \mathsf{n}+3, 1)$-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. The tree "branches" after third round of the protocol where challenge $\mathfrak{z}$ is presented, thus tree T is build by using different values of $\mathfrak{z}$.

We consider two mutually disjunctive events. The first, E holds when all of the transcripts are acceptable by the idealised verification equation, i.e. $\mathsf{ve}_\pi(X) = 0$, cf. Eq. (7). The second, $\overline{\mathsf{E}}$ holds when there is a transcript that is acceptable by the real verifier, but not by the ideal verifier. That is, for that particular transcript holds $\mathsf{ve}_\pi(\chi) = 0$, but $\mathsf{ve}_\pi(X) \neq 0$. When event E holds we construct the extractor $\mathsf{Ext}_{\mathsf{tree}}$. If it does not, we show a reduction $\mathcal{R}_{\mathsf{dlog}}$ that breaks the dlog assumption.

When E happens: Let $(\mathsf{x}, \mathsf{T}) \leftarrow \mathcal{A}(\mathbf{R}, \mathsf{srs})$. Since the protocol $\mathbf{P}$, 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 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 correctly on $(\mathsf{n}+3)$ different challenges $\mathfrak{z}$ then $(\mathsf{n}+3)$ evaluations of these polynomials (at different points) are known. The extractor $\mathsf{Ext}_{\mathsf{tree}}$ interpolates the polynomials and reveals the corresponding witness w. The extractor finds the witness with probability 1.

When $\overline{\mathsf{E}}$ happens: Let $\mathcal{A}$ be an adversary that for relation $\mathbf{R}$ and randomly picked $\mathsf{srs} \leftarrow_\$ \mathsf{KGen}(\mathbf{R})$ produces a tree of acceptable transcripts such that $\overline{\mathsf{E}}$ happens. Let $\mathcal{R}_{\mathsf{dlog}}$ be a reduction that gets as input an $(\mathsf{n}+2, 1)$-dlog instance $[1, \ldots, \chi^{\mathsf{n}}]_1, [\chi]_2$ and is tasked to output χ. The reduction proceeds as follows—it gives the input instance to the adversary as the SRS. Let (x, T) be the output returned by $\mathcal{A}$. Consider a transcript $\pi \in \mathsf{T}$ such that $\mathsf{ve}_\pi(X) \neq 0$, but $\mathsf{ve}_\pi(\chi) = 0$. Since the adversary is algebraic, all group elements included in T are extended by their representation as a combination of the input $\mathbb{G}_1$-elements. Hence all coefficients of the verification equation polynomial $\mathsf{ve}_\pi(X)$ are known and $\mathcal{R}_{\mathsf{dlog}}$ can find its zero points. Since $\mathsf{ve}_\pi(\chi) = 0$, the targeted discrete log value χ is among them. □

5.5 Honest verifier zero-knowledge

Lemma 6. P *is computationally honest verifier zero-knowledge and its simulator* Sim *does not require a SRS trapdoor.*[7] *More precisely, assume that* Plonk*'s SRS simulator* Sim_χ *produces proof that are distributed at most* $\varepsilon_{\mathsf{zk}}$*-far from real proofs, and* $(\mathsf{R}, \mathsf{S}, \mathsf{T}, \mathsf{f}, 1)$*-uber assumption for* $\mathsf{R}, \mathsf{S}, \mathsf{T}, \mathsf{f}$ *as defined in Eq.* (10) *is* $\varepsilon_{\mathsf{uber}}$*-secure. Then any* PPT *adversary* $\mathcal{A}$ *has advantage in telling a proof produced by* Sim *from a real proof upper-bounded by* $\varepsilon_{\mathsf{zk}} + \varepsilon_{\mathsf{uber}}$.

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

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

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

Game G_1: In this game for $\mathcal{A}(\mathbf{R}, \mathsf{srs})$ picks an instance–witness pair (x, w) and gets a proof π that is simulated by a simulator Sim_χ which utilises for the simulation the SRS trapdoor and proceeds as follows. In the first round the simulator Sim_χ picks randomisers $b_1, \dots b_9$, sets $\mathsf{w}_i = 0$, 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 challenges β, γ honestly.

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

In Round 3 the simulator computes polynomial $\mathsf{t}(X)$ and evaluates it at χ, then outputs $[\mathsf{t}_{\mathsf{lo}}(\chi), \mathsf{t}_{\mathsf{mid}}(\chi), \mathsf{t}_{\mathsf{hi}}(\chi)]_1$. Note that this evaluation is feasible (in the polynomial time with non-negligible probability) 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

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

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

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

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

In Round 3 the simulator starts by picking at random a challenge $\mathfrak{z}$, which in the real proof comes as a challenge from the verifier sent *after* Round 3. Then Sim computes evaluations $\mathsf{a}(\mathfrak{z}), \mathsf{b}(\mathfrak{z}), \mathsf{c}(\mathfrak{z}), \mathsf{S}_{\sigma 1}(\mathfrak{z}), \mathsf{S}_{\sigma 2}(\mathfrak{z}), \mathsf{PI}(\mathfrak{z}), \mathsf{L}_1(\mathfrak{z}), \mathsf{Z}_{\mathsf{H}}(\mathfrak{z}), \mathsf{z}(\mathfrak{z}\omega)$ and computes $\mathsf{t}(X)$ honestly. Since for a random $\mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X), \mathsf{z}(X)$ the constraint system is (with overwhelming probability) not satisfied and the constraints-related polynomials are not divisible by $\mathsf{Z}_{\mathsf{H}}(X)$, hence $\mathsf{t}(X)$ is a rational function rather than a polynomial. Then, the simulator evaluates $\mathsf{t}(X)$ at $\mathfrak{z}$ and picks randomly a degree-$(3\mathsf{n}-1)$ polynomial $\tilde{\mathsf{t}}(X)$ such that $\mathsf{t}(\mathfrak{z}) = \tilde{\mathsf{t}}(\mathfrak{z})$ and publishes a commitment $[\tilde{\mathsf{t}}_{\mathsf{lo}}(\chi), \tilde{\mathsf{t}}_{\mathsf{mid}}(\chi), \tilde{\mathsf{t}}_{\mathsf{hi}}(\chi)]_1$. 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. Section 5.1 and evaluates $\mathsf{r}(X)$ at $\mathfrak{z}$.

The rest of the evaluations are already computed, thus Sim simply outputs

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

After that it picks randomly the challenge v, proceeds in the last round as an honest prover would proceed and outputs the final challenge, u, by picking it at random as well.

$\mathsf{G}_1 \mapsto \mathsf{G}_2$: We now describe the reduction $\mathcal{R}$ which relies on the $(\mathsf{R}, \mathsf{S}, \mathsf{T}, \mathsf{F}, 1)$-uber assumption where $\mathsf{R}, \mathsf{S}, \mathsf{T}, \mathsf{F}$ are polynomials over variables $\boldsymbol{B} = B_1, \dots, B_9$ and are defined as follows. Let $E = \{\{2\}, \{3,4\}, \{5,6\}, \{7,8,9\}\}$ and $E' = E \setminus \{2\}$. Let

$$\begin{aligned}
\mathsf{F}(\boldsymbol{B}) &= \{B_1\} \cup \{B_1 B_i \mid i \in A,\ A \in E'\} \cup \{B_1 B_i B_j \mid i \in A, j \in B,\ A, B \in E', B \neq A\} \cup \\
&\quad \{B_1 B_i B_j B_k \mid i \in A, j \in B, k \in C,\ A, B, C \in E', A \neq B \neq C \neq A\}\,, \\
\mathsf{R}(\boldsymbol{B}) &= \{B_i \mid i \in A,\ A \in E\} \cup \{B_i B_j \mid i \in A, j \in B,\ A \neq B, A, B \in E\} \cup \\
&\quad \{B_i B_j B_k \mid i \in A,\ j \in B,\ k \in C,\ A, B, C \text{ all different and in } E\} \cup
\end{aligned} \tag{10}$$

$$\{B_iB_jB_kB_l \mid i \in A,\ j \in B,\ k \in C,\ l \in D,\ A,B,C,D \text{ all different and in } E\}$$
$$\setminus \mathsf{F}(\boldsymbol{B}),$$
$$\mathsf{S}(\boldsymbol{B}) = \emptyset,$$
$$\mathsf{T}(\boldsymbol{B}) = \emptyset.$$

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

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

$$\left[\tilde{\mathsf{b}}_1\right]_1(X) = (X-\mathfrak{z})(X-\omega\mathfrak{z})\,[w_1]_1(X) + \xi_i(X-\mathfrak{z})\,[1]_1 + \zeta_i(X-\omega\mathfrak{z})\,[1]_1,$$

and

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

for $i \in [2\,..\,9]$.

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

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

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

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

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

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

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

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

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

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

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

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

Corollary 1 (Simulation extractability of $\mathbf{P}_{\mathsf{FS}}$). *Assume that $\mathbf{P}$ is 3-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 $\mathcal{A}$ be a PPT adversary that can make up to q random oracle queries and outputs an acceptable proof for $\mathbf{P}_{\mathsf{FS}}$ with probability at least acc. Then $\mathbf{P}_{\mathsf{FS}}$ 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 constrains in the proven circuit.

6 Simulation extractability of $\mathbf{S}_{\mathsf{FS}}$

6.1 Sonic protocol rolled out

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

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

There is also Q linear constrains of the form

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

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

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

In Sonic we will use commitments to the following polynomials.

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

Algorithms rolled out

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

6.2 Unique opening property of $\mathbf{PC_S}$

Lemma 7. $\mathbf{PC_S}$ *has the unique opening property in the AGM.*

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

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

is at most negligible.

As noted in [27, Lemma 2.2] it is enough to upper bound the probability of the adversary succeeding using the idealised verification equation—which considers equality between polynomials—instead of the real verification equation—which considers equality of the polynomials' evaluations.

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

$$\alpha(X^{\mathsf{d}-\mathsf{max}}f(X)\cdot X^{-\mathsf{d}+\mathsf{max}}-s)\equiv\alpha\cdot o(X)(X-z)\,, \tag{12}$$

what is equivalent to

$$f(X)-s\equiv o(X)(X-z)\,. \tag{13}$$

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. □

6.3 Unique response property

The unique response property of $\mathbf{S}$ follows from the unique opening property of the used polynomial commitment scheme $\mathbf{PC}_{\mathbf{S}}$.

Lemma 8. *If a polynomial commitment scheme $\mathbf{PC}_{\mathbf{S}}$ is evaluation binding with parameter $\varepsilon_{\mathsf{bind}}$ and has unique openings property with parameter $\varepsilon_{\mathsf{op}}$, then $\mathbf{S}$ is 2-ur with parameter $\varepsilon_{\mathsf{ur}}\leq\varepsilon_{\mathsf{bind}}+\varepsilon_{\mathsf{op}}$.*

Proof. Let $\mathcal{A}$ be an adversary that breaks 2-ur-ness of $\mathbf{S}$. We consider two cases, depending on which round $\mathcal{A}$ is able to provide at least two different outputs such that the resulting transcripts are acceptable. For the first case we show that $\mathcal{A}$ can be used to break the evaluation binding property of $\mathbf{PC}_{\mathbf{S}}$, while for the second case we show that it can be used to break the unique opening property of $\mathbf{PC}_{\mathbf{S}}$.

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 randomly picked points, or shows that the evaluations were performed correctly. Obviously, for a committed polynomial p evaluated at point x only one value $y=\mathsf{p}(x)$ is correct. If the adversary was able to provide two different values y and $\tilde{y}$ that would be accepted as an evaluation of p at x then the $\mathbf{PC}_{\mathbf{S}}$'s evaluation binding would be broken. Alternatively, if $\mathcal{A}$ was able to provide two openings W and $\tilde{\mathsf{W}}$ for $y=\mathsf{p}(x)$ then the unique opening property would be broken. Hence the probability that $\mathcal{A}$ breaks 2-ur-property of $\mathbf{PC}_{\mathbf{S}}$ is upper-bounded by $\varepsilon_{\mathsf{bind}}+\varepsilon_{\mathsf{op}}$. □

6.4 Computational special soundness

Lemma 9. *Let $\mathcal{A}$ be a* PPT *algebraic adversary. The probability $\varepsilon_{\mathsf{ss}}$ that $\mathcal{A}$ breaks computational special soundness of $\mathbf{S}$ is upper-bounded as*

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

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

Proof. The proof goes similarly to the proof of Lemma 5. Let $\mathcal{A}$ be an adversary that produces a $(1,\mathsf{n}+\mathsf{Q}+1,1,1)$-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. $\mathbf{ve}(X)=\mathbf{0}$. In that case we show an extractor $\mathsf{Ext}_{\mathsf{ss}}$ 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 (d,d)-ldlog assumption.

When E happens: Since $\mathbf{S}$ is statistically 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 polynomial $\mathsf{r}(X,y)$, which has witness elements at its coefficients, has degree at most $\mathsf{n}+\mathsf{Q}$ and since $\mathcal{A}$ answered correctly on $(\mathsf{n}+\mathsf{Q}+1)$ different challenges z (for the sake of concreteness let us call them $z_1,\ldots,z_{\mathsf{n}+\mathsf{Q}+1}$) 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. □

6.5 Honest verifier zero-knowledge

Lemma 10. 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

$$\boldsymbol{c} = \boldsymbol{a} \cdot \boldsymbol{b}. \tag{14}$$

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 the polynomial $\mathsf{r}(X,Y)$ computed by the simulator is indistinguishable from a polynomial provided by an honest user for an adversary that only learns $[r]_1, [t]_1, a', b'$ from the proof (the other proof elements are fixed by these elements and the public information, see Section 6.1).

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. $\mathsf{t}(0,y) = 0$. By the definition of $\mathsf{t}(X,Y)$, the coefficient next to X^0 in $\mathsf{t}(X,Y)$ equals

$$\mathsf{t}(0,Y) = \boldsymbol{a} \cdot \mathbf{u}(Y) + \boldsymbol{b} \cdot \mathbf{v}(Y) + \boldsymbol{c} \cdot \mathbf{w}(Y) + \sum_{i=1}^{\mathsf{n}} a_i b_i (Y^i + Y^{-i}) - \mathsf{k}(Y), \tag{15}$$

for public $\mathbf{u}(Y), \mathbf{v}(Y), \mathbf{w}(Y), \mathsf{k}(Y)$ as defined in Section 6.1 (Vectors $\boldsymbol{u_q}, \boldsymbol{v_q}, \boldsymbol{w_q}$ are n-elements long and correspond to 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 [40] for details. Also, when Eq. (14) holds, that polynomial simplifies to

$$\mathsf{t}(0,Y) = \boldsymbol{a} \cdot \mathbf{u}(Y) + \boldsymbol{b} \cdot \mathbf{v}(Y) + \boldsymbol{c} \cdot \tilde{\mathbf{w}}(Y) - \mathsf{k}(Y), \tag{16}$$

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 $\mathsf{t}(0,Y)$ is a "classical", i.e. non-Laurent, polynomial. Also, it is a polynomial of degree $\mathsf{Q}+\mathsf{n}$ with 0 coefficients for Y^i, for $i \in [0\,..\,\mathsf{n}]$. 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 $\mathbb{F}_p$ elements, 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 [40], Sonic is statistically subversion-zero knowledge (Sub-ZK). As noted in [1], one way to achieve subversion zero knowledge is to utilise an extractor that extracts a SRS trapdoor from a SRS-generator. Unfortunately, a NIZK made subversion zero-knowledge by this approach cannot achieve perfect Sub-ZK as one has to count in the probability of extraction failure. However, with the simulation presented in Lemma 10, 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 $\mathbf{S}_{\mathsf{FS}}$, where the simulator can program the random oracle to provide challenges that fits it.

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

Since Lemmas 8 and 9 hold, $\mathbf{S}$ is 2-ur and computationally special sound. We now make use of Theorem 1 and show that $\mathbf{S}_{\mathsf{FS}}$ is simulation-extractable as defined in Definition 5.

Corollary 2 (Simulation extractability of S_{FS}). *Assume that $\mathbf{S}$ 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 $\mathcal{A}$ be a PPT adversary that can make up to q random oracle queries and outputs an acceptable proof for $\mathbf{S}_{\mathsf{FS}}$ with probability at least acc. Then $\mathbf{S}_{\mathsf{FS}}$ is simulation-extractable with extraction error $\eta = \varepsilon_{\mathsf{ur}}$. The extraction probability ext is at least*

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

for some negligible ε, n and Q being, respectively, the number of multiplicative and linear constrains of the system.

7 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 SRS-based NIZKs. Thus, it would be interesting to apply it to known so-called transparent zkSNARKs like Bulletproofs [14], Aurora [9] or AuroraLight [26].

Since the rewinding technique and the forking lemma used to show simulation extractability of $\mathbf{P}_{\mathsf{FS}}$ and $\mathbf{S}_{\mathsf{FS}}$ 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.

Acknowledgement

The second author thanks Antoine Rondelet for helpful discussions.

References

1. B. Abdolmaleki, K. Baghery, H. Lipmaa, and M. Zajac. A subversion-resistant SNARK. In T. Takagi and T. Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part III*, volume 10626 of *Lecture Notes in Computer Science*, pages 3–33. Springer, Heidelberg, Dec. 2017.
2. B. Abdolmaleki, S. Ramacher, and D. Slamanig. Lift-and-shift: Obtaining simulation extractable subversion and updatable SNARKs generically. In J. Ligatti, X. Ou, J. Katz, and G. Vigna, editors, *ACM CCS 20: 27th Conference on Computer and Communications Security*, pages 1987–2005. ACM Press, Nov. 2020.
3. S. Atapoor and K. Baghery. Simulation extractability in groth's zk-SNARK. Cryptology ePrint Archive, Report 2019/641, 2019. `https://eprint.iacr.org/2019/641`.
4. Aztec. A private layer 2. `https://developers.aztec.network`, 2020.
5. K. Baghery, M. Kohlweiss, J. Siim, and M. Volkhov. Another look at extraction and randomization of groth's zk-SNARK. Cryptology ePrint Archive, Report 2020/811, 2020. `https://eprint.iacr.org/2020/811`.
6. M. Bellare, G. Fuchsbauer, and A. Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. In J. H. Cheon and T. Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 777–804. Springer, Heidelberg, Dec. 2016.
7. M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In A. Juels, R. N. Wright, and S. De Capitani di Vimercati, editors, *ACM CCS 2006: 13th Conference on Computer and Communications Security*, pages 390–399. ACM Press, Oct. / Nov. 2006.

8. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, editors, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, Nov. 1993.
9. E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. Aurora: Transparent succinct arguments for R1CS. In Y. Ishai and V. Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 103–128. Springer, Heidelberg, May 2019.
10. D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer, Heidelberg, May 2005.
11. J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In M. Fischlin and J.-S. Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 327–357. Springer, Heidelberg, May 2016.
12. S. Bowe and A. Gabizon. Making groth's zk-SNARK simulation extractable in the random oracle model. Cryptology ePrint Archive, Report 2018/187, 2018. `https://eprint.iacr.org/2018/187`.
13. X. Boyen. The uber-assumption family (invited talk). In S. D. Galbraith and K. G. Paterson, editors, *PAIRING 2008: 2nd International Conference on Pairing-based Cryptography*, volume 5209 of *Lecture Notes in Computer Science*, pages 39–56. Springer, Heidelberg, Sept. 2008.
14. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.
15. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. `https://eprint.iacr.org/2000/067`.
16. A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In A. Canteaut and Y. Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 738–768. Springer, Heidelberg, May 2020.
17. cLabs. The celo protocol: A multi-asset cryptographic protocol fordecentralized social payments. `https://celo.org/papers/whitepaper`, 2020.
18. Clearmatics. Zeth: On integrating zerocash on ethereum. `https://www.github.com/clearmatics/zeth`, 2020.
19. G. Danezis, C. Fournet, J. Groth, and M. Kohlweiss. Square span programs with applications to succinct NIZK arguments. In P. Sarkar and T. Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 532–550. Springer, Heidelberg, Dec. 2014.
20. A. W. Dent. A note on game-hopping proofs. Cryptology ePrint Archive, Report 2006/260, 2006. `https://eprint.iacr.org/2006/260`.
21. Y. Dodis, K. Haralambiev, A. López-Alt, and D. Wichs. Efficient public-key cryptography in the presence of key leakage. In M. Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 613–631. Springer, Heidelberg, Dec. 2010.
22. S. Faust, M. Kohlweiss, G. A. Marson, and D. Venturi. On the non-malleability of the Fiat-Shamir transform. In S. D. Galbraith and M. Nandi, editors, *Progress in Cryptology - INDOCRYPT 2012: 13th International Conference in Cryptology in India*, volume 7668 of *Lecture Notes in Computer Science*, pages 60–79. Springer, Heidelberg, Dec. 2012.
23. M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In V. Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 152–168. Springer, Heidelberg, Aug. 2005.
24. G. Fuchsbauer. Subversion-zero-knowledge SNARKs. In M. Abdalla and R. Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10769 of *Lecture Notes in Computer Science*, pages 315–347. Springer, Heidelberg, Mar. 2018.
25. G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In H. Shacham and A. Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 33–62. Springer, Heidelberg, Aug. 2018.
26. A. Gabizon. On the security of the BCTV pinocchio zk-SNARK variant. Cryptology ePrint Archive, Report 2019/119, 2019. `https://eprint.iacr.org/2019/119`.
27. A. Gabizon, Z. J. Williamson, and O. Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. `https://eprint.iacr.org/2019/953`.
28. R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct NIZKs without PCPs. In T. Johansson and P. Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 626–645. Springer, Heidelberg, May 2013.
29. S. Goldwasser and Y. T. Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th Annual Symposium on Foundations of Computer Science*, pages 102–115. IEEE Computer Society Press, Oct. 2003.
30. J. Groth. Fully anonymous group signatures without random oracles. In K. Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 164–180. Springer, Heidelberg, Dec. 2007.
31. J. Groth. Short pairing-based non-interactive zero-knowledge arguments. In M. Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 321–340. Springer, Heidelberg, Dec. 2010.
32. J. Groth. On the size of pairing-based non-interactive arguments. In M. Fischlin and J.-S. Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 305–326. Springer, Heidelberg, May 2016.
33. J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn, and I. Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In H. Shacham and A. Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 698–728. Springer, Heidelberg, Aug. 2018.

34. J. Groth and M. Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In J. Katz and H. Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 581–612. Springer, Heidelberg, Aug. 2017.
35. A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In M. Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 177–194. Springer, Heidelberg, Dec. 2010.
36. A. Kosba, Z. Zhao, A. Miller, Y. Qian, H. Chan, C. Papamanthou, R. Pass, a. shelat, and E. Shi. How to use SNARKs in universally composable protocols. Cryptology ePrint Archive, Report 2015/1093, 2015. `https://eprint.iacr.org/2015/1093`.
37. H. Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In R. Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 169–189. Springer, Heidelberg, Mar. 2012.
38. H. Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In K. Sako and P. Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 41–60. Springer, Heidelberg, Dec. 2013.
39. H. Lipmaa. Simulation-extractable SNARKs revisited. Cryptology ePrint Archive, Report 2019/612, 2019. `https://eprint.iacr.org/2019/612`.
40. M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In L. Cavallaro, J. Kinder, X. Wang, and J. Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 2111–2128. ACM Press, Nov. 2019.
41. B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013.
42. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000.
43. A. Rondelet and M. Zajac. Zeth: On integrating zerocash on ethereum. https://arxiv.org/abs/1904.00905, 2019.
44. C.-P. Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, Heidelberg, Aug. 1990.
45. V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, Heidelberg, May 1997.
46. V. Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. `https://eprint.iacr.org/2004/332`.
47. E. Weisstein. Jensen's inequality.
48. Zcash. Zchas documentation. `https://zcash.readthedocs.io`, 2020.

A Simulation-extractability of sigma protocols and forking lemma

Theorem 2 (Simulation extractability of the Fiat–Shamir transform [22]). *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* [42].

Lemma 11 (General forking lemma, cf. [7, 22]). *Fix $q \in \mathbb{Z}$ and a set H of size $h > 2$. Let $\mathcal{Z}$ be a PPT algorithm that on input $y, h_1, \ldots, h_q$ returns (i, s), where $i \in [0\,..\,q]$ and s is called a* side output*. Denote by IG a randomised instance generator. We denote by acc the probability*

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

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

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

B 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 $\mathbb{E}[X_\iota^m] \geq \mathbb{E}[X_\iota]^m$.

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

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

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

Proof. First we recall the Jensen inequality [47], if for some random variable X holds $|\mathbb{E}[X]| \leq \infty$ and f is a Borel convex function then

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

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

Lemma 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 $\mathbb{E}[X_i] = x$, $i \in [1 .. q]$ we have

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

Lemma 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

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

□

B.1 Uber assumption

We show security of our version of the uber assumption using the generic group model as introduced by Shoup [45] where all group elements are represented by random binary strings of length λ. That is, there are random encodings ξ_1, ξ_2, ξ_T which are injective functions from $\mathbb{Z}_p^+$ to $\{0,1\}^\lambda$. We write $\mathbb{G}_i = \{\xi_i(x) \mid x \in \mathbb{Z}_p^+\}$, for $i \in \{1, 2, T\}$. For the sake of clarity we denote by $\xi_{i,j}$ the j-th encoding in group $\mathbb{G}_i$.

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

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

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

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

Given that, we are ready to show security of our variant of the Boneh et al. uber assumption. The proof goes similarly to the original proof given in [10] with minor differences.

Theorem 3 (Security of the uber assumption). *Let $\mathsf{P}_i \in \mathbb{F}_p[X_1, \ldots, X_n]^{m_i}$, for $i \in \{1, 2, T\}$ be τ_i tuples of n-variate polynomials over $\mathbb{F}_p$ and let $\mathsf{F} \in \mathbb{F}_p[X_1, \ldots, X_n]^m$. Let ξ_0, ξ_1, ξ_T, $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be as defined above. If polynomials $f \in \mathsf{F}$ are pair-wise independent and are independent of $\mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_T$, then for any $\mathcal{A}$ that makes up to q queries to the GGM oracle holds:*

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Finally,

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

as required.

C (Tight) simulation soundness of Plonk

Theorem 4 (Simulation soundness). *Assume that* $(\mathsf{n}+2,1)$-dlog *is* $\varepsilon_{\mathsf{dlog}}(\lambda)$*-hard,* $\mathbf{P}$ *is sound and* 2-ur *with security* $\varepsilon_{\mathsf{s}}(\lambda)$ *and* $\varepsilon_{\mathsf{ur}}(\lambda)$ *respectively. Then the probability that an algebraic,* PPT *adversary* $\mathcal{A}_{\mathsf{ss}}$ *breaks simulation soundness of* $\mathbf{P}_{\mathsf{FS}}$ *is upper-bounded by*

$$\varepsilon_{\mathsf{ur}}(\lambda) + q_{\mathcal{H}}^6(\varepsilon_{\mathsf{dlog}}(\lambda) + \varepsilon_{\mathsf{s}}(\lambda)),$$

where $q_{\mathcal{H}}$ *is the total number of queries required by the adversary* $\mathcal{A}_{\mathsf{ss}}$.

Proof. We proceed by contradiction. Suppose there exists a PPT adversary $\mathcal{A}_{\mathsf{ss}}$ that breaks simulation soundness with non-negligible probability

$$\varepsilon := \Pr\left[\begin{array}{l|l} \mathbf{P}_{\mathsf{FS}}.\mathsf{V}(\mathbf{R},\mathsf{srs},\mathsf{x},\pi_{\mathcal{A}_{\mathsf{ss}}}), & \mathsf{srs} \leftarrow \mathbf{P}_{\mathsf{FS}}.\mathsf{KGen}(\mathbf{R},1^\lambda) \\ (\mathsf{x}_{\mathcal{A}_{\mathsf{ss}}},\pi_{\mathcal{A}_{\mathsf{ss}}}) \notin Q, & (\mathsf{x}_{\mathcal{A}_{\mathsf{ss}}},\pi_{\mathcal{A}_{\mathsf{ss}}}) \leftarrow \mathcal{A}_{\mathsf{ss}}^{\mathsf{Sim},\mathcal{H}}(\mathbf{R},\mathsf{srs}), \\ \mathsf{x}_{\mathcal{A}_{\mathsf{ss}}} \notin \mathcal{L}_{\mathbf{R}} & \end{array}\right].$$

In such case, we are able to build reductions $\mathcal{R}_{\mathsf{s}}$, $\mathcal{R}_{\mathsf{ur}}$, $\mathcal{R}_{\mathsf{dlog}}$ which using $\mathcal{A}_{\mathsf{ss}}$ as a black-box, violate either the soundness, unique response properties of the underlying interactive protocol $\mathbf{P}$, or the $(\mathsf{n}+2,1)$-dlog assumption.

In the following we denote by $\pi_{\mathcal{A}_{\mathsf{ss}}}, \pi_{\mathsf{Sim}}$ 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, $\pi[i].\mathsf{ch}$ to denote 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.

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.

A crucial observation is that the adversary $\mathcal{A}_{\mathsf{ss}}$ may have learned $\pi_{\mathcal{A}_{\mathsf{ss}}}[1\,..\,3]$ by querying the simulator on $\mathsf{x}_{\mathcal{A}_{\mathsf{ss}}}$ or might have computed it itself. We denote the first event by E and the second by $\overline{\mathsf{E}}$. Additionally, we divide event $\overline{\mathsf{E}}$ into two disjunctive subevents: $\overline{\mathsf{E}}_0$ and $\overline{\mathsf{E}}_1$. Event $\overline{\mathsf{E}}_0$ considers a case when the final proof provided by the adversary $\mathcal{A}_{\mathsf{ss}}$ is accepted by the idealised verification equation, i.e. for that proof $\mathsf{ve}_\pi(X) = 0$. Alternatively, event $\overline{\mathsf{E}}_1$ covers a case when for $\pi_{\mathcal{A}_{\mathsf{ss}}}$ it holds that $\mathsf{ve}_\pi(\chi) = 0$, but $\mathsf{ve}_\pi(X) \neq 0$, where χ is $\mathbf{P}_{\mathsf{FS}}$'s trapdoor. As all these events are mutually exclusive and exhaustive, we have

$$\varepsilon = \Pr[\mathcal{A}_{\mathsf{ss}} \text{ wins}] = \Pr[\mathcal{A}_{\mathsf{ss}} \text{ wins}, \mathsf{E}] + \Pr\left[\mathcal{A}_{\mathsf{ss}} \text{ wins}, \overline{\mathsf{E}}_0\right] + \Pr\left[\mathcal{A}_{\mathsf{ss}} \text{ wins}, \overline{\mathsf{E}}_1\right] .$$

Before analysing the events, we make the following observation. First of all, we allow reductions $\mathcal{R}_{\mathsf{dlog}}, \mathcal{R}_{\mathsf{ur}}, \mathcal{R}_{\mathsf{s}}$ to simulate the random oracle and simulator for the adversary $\mathcal{A}_{\mathsf{ss}}$. We argue that since the reductions in their simulation behaves as real random oracle or simulator would, the chances the adversary breaks simulation soundness does not change.

Furthermore, we note that since $\mathcal{A}_{\mathsf{ss}}$ is algebraic, it outputs a proof $\pi_{\mathcal{A}_{\mathsf{ss}}}$ that can be written as

$$\pi_{\mathcal{A}_{\mathsf{ss}}} = \boldsymbol{M} \cdot (\underbrace{1 \| \chi \| \ldots \| \chi^{\mathsf{n}+2}}_{\mathsf{srs}} \| \tilde{\boldsymbol{\pi}}_{\mathsf{Sim}1}^{\top} \| \ldots \| \tilde{\boldsymbol{\pi}}_{\mathsf{Sim}q_{\mathsf{Sim}}}^{\top})^{\top} ,$$

where $[1, \chi, \ldots, \chi^{\mathsf{n}+2}]_1$ are all $\mathbb{G}_1$-elements from the SRS of $\mathbf{P}_{\mathsf{FS}}$, $\boldsymbol{M}$ is a matrix of coefficients output by $\mathcal{A}_{\mathsf{ss}}$ aside the proof, $\tilde{\boldsymbol{\pi}}_{\mathsf{Sim}i}$ denote all $\mathbb{G}_1$-elements from the simulated proof $\pi_{\mathsf{Sim}i}$, and the adversary makes q_{Sim} queries to the simulator. Since the reduction itself provides the simulated proofs, it knows a matrix $\boldsymbol{M'}$ such that

$$\pi_{\mathcal{A}_{\mathsf{ss}}} = \boldsymbol{M'} \cdot (1 \| \chi \| \ldots \| \chi^{\mathsf{n}+2})^{\top} . \tag{17}$$

We use this property when analysing the success probability of reductions $\mathcal{R}_{\mathsf{s}}$ and $\mathcal{R}_{\mathsf{dlog}}$.

Also note that a proof π could be accepted only if the verification equation $\mathsf{ve}_\pi(\chi)$ holds. That is, the verifier plugs-in elements of π into $\mathsf{ve}_\pi(\chi)$ and checks whether it equals 0. That is what is called a *real check* in [27]. On the other hand there is an *idealised check*, which verifies whether $\mathsf{ve}_\pi(X) = 0$ *as a polynomial*—with proof elements being polynomials as well.

When E happens: We assume that $\mathsf{x}_{\mathcal{A}_{\mathsf{ss}}}$ is submitted to the simulator Sim. We show how $\mathcal{R}_{\mathsf{ur}}$ utilizes $\mathcal{A}_{\mathsf{ss}}$, that makes use of $\mathsf{x}_{\mathcal{A}_{\mathsf{ss}}}, \pi_{\mathcal{A}_{\mathsf{ss}}}[1\,..\,3]$, to break the 2-$\mathsf{ur}$ property of $\mathbf{P}$. This way we bound the probability $\Pr[\mathcal{A} \text{ wins}, \mathsf{E}]$ by the probability of $\mathcal{R}_{\mathsf{ur}}$ being able to win in the 2-ur game.

Consider an algorithm $\mathcal{R}_{\mathsf{ur}}$ that runs $\mathcal{A}_{\mathsf{ss}}$ internally as a black-box:

- The reduction answers both queries to the simulator $\mathbf{P}_{\mathsf{FS}}.\mathsf{Sim}$ and to the random oracle. It also keeps lists Q, for the simulated proofs, and $Q_{\mathcal{H}}$ for the random oracle queries.
- When $\mathcal{A}_{\mathsf{ss}}$ outputs a fake proof $\pi_{\mathcal{A}_{\mathsf{ss}}}$ for $\mathsf{x}_{\mathcal{A}_{\mathsf{ss}}}$, $\mathcal{R}_{\mathsf{ur}}$ looks through lists Q and $Q_{\mathcal{H}}$ until it finds $\pi_{\mathsf{Sim}}[1\,..\,3]$ such that $\pi_{\mathcal{A}_{\mathsf{ss}}}[1\,..\,2] = \pi_{\mathsf{Sim}}[1\,..\,3]$ and a random oracle query $\pi_{\mathsf{Sim}}[3].\mathsf{ch}$ on $\pi_{\mathsf{Sim}}[1\,..\,3]$.
- $\mathcal{R}_{\mathsf{ur}}$ returns two proofs for $\mathsf{x}_{\mathcal{A}_{\mathsf{ss}}}$:

$$\pi_1 = (\pi_{\mathsf{Sim}}[1\,..\,3], \pi_{\mathsf{Sim}}[3].\mathsf{ch}, \pi_{\mathsf{Sim}}[4\,..\,5])$$
$$\pi_2 = (\pi_{\mathsf{Sim}}[1\,..\,3], \pi_{\mathsf{Sim}}[3].\mathsf{ch}, \pi_{\mathcal{A}_{\mathsf{ss}}}[4\,..\,5])$$

If $\pi_1 = \pi_2$, then $\mathcal{A}_{\mathsf{ss}}$ 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 2-ur-ness of $\mathbf{P}$. Thus

$$\Pr[\mathcal{A}_{\mathsf{ss}} \text{ wins}, \mathsf{E}] \leq \varepsilon_{\mathsf{ur}}(\lambda).$$

When $\overline{\mathsf{E}}_0$ happens: In this case the reduction $\mathcal{R}_{\mathsf{s}}$ uses $\mathcal{A}_{\mathsf{ss}}$ to break soundness of $\mathbf{P}$ with probability $\varepsilon_{\mathsf{s}}/q_{\mathcal{H}}^6$, where $q_{\mathcal{H}}$ is the number of total random oracle queries performed by the adversary or by $\mathcal{R}_{\mathsf{s}}$ on

behalf of the simulator. As previously, $\mathcal{R}_{\mathsf{s}}$ runs $\mathcal{A}_{\mathsf{ss}}$ internally and simulates its environment by answering to its queries to $\mathbf{P}_{\mathsf{FS}}.\mathsf{Sim}$ and $\mathcal{H}$. The reduction works as follows:

- It guesses indices $i_1, \ldots, i_6$ such that random oracle queries $h_{i_1}, \ldots, h_{i_6}$ are the queries used in $\pi_{\mathcal{A}_{\mathsf{ss}}}$. This is done with probability at least $1/q_{\mathcal{H}}^6$ (since there are 6 challenges from the verifier in $\mathbf{P}$).
- On input h for the i-th, $i \notin \{i_1, \ldots, i_6\}$, random oracle query, $\mathcal{R}_{\mathsf{s}}$ returns randomly picked y, sets $\mathcal{H}(h) = y$ and stores (h, y) in $Q_{\mathcal{H}}$ if h is sent to $\mathcal{H}$ the first time. If that is not the case, $\mathcal{R}$ finds h in $Q_{\mathcal{H}}$ and returns the corresponding y.
- On input h_{i_j} for the i_j-th, $i_j \in \{i_1, \ldots, i_6\}$, random oracle query, $\mathcal{R}_{\mathsf{s}}$ parses h_{i_j} as a partial proof transcript $\pi[1..j]$ and runs $\mathbf{P}$ using $\pi[j]$ as a $\mathbf{P}$.P's j-th message to $\mathbf{P}$.V. The verifier responds with a challenge $\pi[j].\mathsf{ch}$. The reduction sets $\mathcal{H}(h_{i_j}) = \pi[j].\mathsf{ch}$.
- On query $\mathsf{x}_{\mathsf{Sim}}$ to Sim it runs a simulator $\mathbf{P}_{\mathsf{FS}}.\mathsf{Sim}$ internally and returns π_{Sim}. If the random oracle query with input $\pi_{\mathsf{Sim}}[j]$, $1 \leq j \leq 2$, of the simulator is the i_j-th query, generate $\pi_{\mathsf{Sim}}[j].\mathsf{ch}$ by invoking $\mathbf{P}$.V on $\pi_{\mathsf{Sim}}[j]$ and programming $\mathcal{H}(h_{i_j}) = \pi_{\mathsf{Sim}}[j].\mathsf{ch}$.
- Answers $\mathbf{P}$.V's challenge $\pi[j].\mathsf{ch}$ using the answer given by $\mathcal{A}_{\mathsf{ss}}$, i.e. $\pi_{\mathcal{A}_{\mathsf{ss}}}[j+1]$.

Assume that the $\mathbf{P}$.V accepts $\pi_{\mathcal{A}_{\mathsf{ss}}}$. We consider a case when the idealised verification equation accepts. (Thus, the real verification accepts as well.) In that case $\mathcal{R}_{\mathsf{s}}$ extracts from $\boldsymbol{M'}$ coefficients of $1, \chi, \ldots, \chi^{\mathsf{n}+2}$ for polynomials $\mathsf{a}(X), \mathsf{b}(X)$, and $\mathsf{c}(X)$ and reveals the witness $\mathsf{w}_{\mathcal{A}_{\mathsf{ss}}}$ (as it is encoded in theses polynomials' coefficients). If $\mathbf{R}(\mathsf{x}_{\mathcal{A}_{\mathsf{ss}}}, \mathsf{w}_{\mathcal{A}_{\mathsf{ss}}})$ holds then $\mathcal{A}_{\mathsf{ss}}$ failed to break simulation-soundness of $\mathbf{P}_{\mathsf{FS}}$. On the other hand, if that is not the case, then $\mathcal{R}_{\mathsf{s}}$ breaks soundness of $\mathbf{P}$. Since the reduction guesses queries $h_{i_1}, \ldots, h_{i_6}$ with probability $1/q_{\mathcal{H}}^6$, then

$$\Pr[\mathcal{R}_{\mathsf{s}} \text{ wins}] = \Pr\left[\mathcal{A}_{\mathsf{ss}} \text{ wins}, h_{i_1}, \ldots, h_{i_6} \text{ are guessed correctly}, \overline{\mathsf{E}}_0\right].$$

Hence,

$$\Pr\left[\mathcal{A}_{\mathsf{ss}} \text{ wins}, \overline{\mathsf{E}}_0\right] \leq q_{\mathcal{H}}^6 \cdot \varepsilon_{\mathsf{ss}}(\lambda).$$

When $\overline{\mathsf{E}}_1$ happens: The reduction $\mathcal{R}_{\mathsf{dlog}}$ runs internally a protocol $\mathbf{P}_{\mathsf{FS}}$, which SRS is computed from the challenge $\left[1, \chi, \ldots, \chi^{\mathsf{n}+2}\right]_1, [\chi]_2$ from the $(\mathsf{n}+2, 1)$-dlog assumption challenger. Then it proceeds as $\mathcal{R}_{\mathsf{s}}$ does, except in the last part, when the adversary provided its proof $\pi_{\mathcal{A}_{\mathsf{ss}}}$, $\mathcal{R}_{\mathsf{dlog}}$ uses the fact that the real verification equation holds, but the ideal verification equation does not to break the dlog assumption.

Since $\mathsf{ve}_\pi(X) \neq 0$, but $\mathsf{ve}_\pi(\chi) = 0$ and $\mathcal{R}_{\mathsf{dlog}}$ knows $\boldsymbol{M'}$, as defined in Eq. (17), it can recreate all the polynomials submitted by $\mathcal{A}_{\mathsf{ss}}$ as part of the proof and included in $\mathsf{ve}_\pi(X)$. This way, it knows all coefficients of $\mathsf{ve}_\pi(X)$. Thus it can factorize it and find its roots, one of them is the required χ. Hence it holds, by the analogous analysis as in the previous case, that

$$\Pr\left[\mathcal{A}_{\mathsf{ss}} \text{ wins}, \overline{\mathsf{E}}_1\right] \leq q_{\mathcal{H}}^6 \cdot \varepsilon_{\mathsf{dlog}}(\lambda).$$

The proof is concluded by observing that the analysis of events $\mathsf{E}, \overline{\mathsf{E}}_0, \overline{\mathsf{E}}_1$ gives

$$\varepsilon \leq \varepsilon_{\mathsf{ur}}(\lambda) + q_{\mathcal{H}}^6(\varepsilon_{\mathsf{dlog}}(\lambda) + \varepsilon_{\mathsf{ss}}(\lambda)),$$

hence ε is negligible if dlog is hard and $\mathbf{P}$ is sound and 2-ur. □