Fiat-Shamir Bulletproofs are Non-Malleable (in the Random Oracle Model)[⋆]

Chaya Ganesh[1], Claudio Orlandi[2], Mahak Pancholi[2], Akira Takahashi[4], and Daniel Tschudi[4]

[1] Indian Institute of Science
chaya@iisc.ac.in
[2] Aarhus University
{orlandi,mahakp}@cs.au.dk
[3] University of Edinburgh
takahashi.akira.58s@gmail.com
[4] Concordium
dt@concordium.com

February 8, 2023

Abstract. Bulletproofs (Bünz et al. IEEE S&P 2018) are a celebrated ZK proof system that allows for short and efficient proofs, and have been implemented and deployed in several real-world systems. In practice, they are most often implemented in their non-interactive version obtained using the Fiat-Shamir transform. A security proof for this setting is necessary for ruling out malleability attacks. These attacks can lead to very severe vulnerabilities, as they allow an adversary to forge proofs re-using or modifying parts of the proofs provided by the honest parties. An earlier version of this work (Ganesh et al. EUROCRYPT 2022) provided evidence for non-malleability of Fiat-Shamir Bulletproofs. This was done by proving simulation-extractability, which implies non-malleability, in the algebraic group model. In this work, we generalize the former result and prove simulation extractability in the programmable random oracle model, removing the need for the algebraic group model. Along the way, we establish a generic chain of reductions for Fiat-Shamir-transformed multi-round public-coin proofs to be simulation-extractable in the (programmable) random oracle model, which may be of independent interest.

[⋆] An extended abstract appeared at EUROCRYPT 2022. This is a full version of [GOP+22] with additional improved results and supersedes [GOP+21].

Table of Contents

1 Introduction

Zero-knowledge (ZK) proof systems [GMR85] are one of the most fascinating ideas in modern cryptography, as they allow a prover to persuade a verifier that some statement is true without revealing any other information. In recent years we have observed a new renaissance for ZK proofs, motivated in large part by their applications to advanced Blockchain applications. This has led, among other things, to a standardization effort for ZK proofs.[5]

A celebrated modern ZK proof system is Bulletproofs [BBB+18]. Bulletproofs offer transparent setup, short proofs and efficient verification (and it is therefore a *zero-knowledge succinct argument of knowledge* or zkSNARK) using only very well established computational assumptions, namely the hardness of discrete logarithms. At the heart of Bulletproofs lies an "inner product" component. This can be used then for general purpose proofs (i.e., where the statement is described as an arithmetic circuit) or for specific purpose proofs (i.e., range proofs, which are the most common use case in practice). Bulletproofs have been implemented in real world systems, especially for confidential transaction systems, like Monero, Mimblewimble, MobileCoin, Interstellar, etc.

Most practical applications of Bulletproofs utilize their non-interactive variant which, since Bulletproofs is a public-coin proof system, can be obtained using the Fiat-Shamir heuristic [FS87] e.g., the interaction with the verifier (who is only supposed to send uniformly random challenges) is replaced by interacting with a public hash function. Under the assumption that the hash function is a random oracle, one can hope that the prover has no easier time producing proofs for false statements (or for statements for which they do not know a witness) than when interacting with an actual verifier.

While the Fiat-Shamir heuristic does not always guarantee soundness in the standard model [GK03, BDG+13], the programmable random oracle model allows to construct sound non-interactive zero knowledge proofs and digital signatures [OO98, PS00, AABN02, FKMV12, KMP16]. However, most existing results only apply to classic Σ-protocols [CDS94], which are a special class of ZK protocols with only 3 moves. Therefore these analyses do not cover the case of Bulletproofs, which is a multi-round protocol.

For the case of Bulletproofs, Ghoshal and Tessaro [GT21] conducted a concrete knowledge soundness analysis of Fiat-Shamir Bulletproofs in the *algebraic group model* [FKL18], e.g., it is not possible for any algebraic prover to produce a valid proof without knowing a witness for the statement (a similar result, but with less tight bounds, appeared concurrently also in [BMM+19]). Later, a recent work of Attema, Fehr and Klooß [AFK21] formally proved the *assumed* concrete knowledge error of Fiat–Shamir transformation of Bulletproofs-like protocols [6] in the random oracle model only. However, the results in [GT21], [AFK21] only consider a malicious prover "in isolation", whereas in most practical applications of Bulletproofs, several provers are producing and exchanging proofs at the same time (e.g., on a Blockchain).

The notion of *non-malleability* in cryptography was introduced in [DDN91], and the notion of non-malleability for zero-knowledge proofs was introduced in [Sah99]. In a nutshell, a malleability attack is one in which the adversary gets to see proofs from honest parties, and then modifies or re-uses parts of the proofs output by the honest parties to forge a proof on some statement for which they do not know a witness. Malleability attacks can have very serious consequences, such as the famous MtGox attack of 2014 [DW14].

Therefore, it is worrisome that Fiat-Shamir Bulletproofs have been implemented in the wild without any solid evidence that malleability attacks are not possible against them.

Improvements over the proceeding version [GOP+22]. In the previous version [GOP+22], we show that Fiat-Shamir Bulletproofs satisfies a strong notion of *simulation-extractability* which in particular implies *non-malleability.* The result assumes the algebraic group model (AGM) [FKL18] which is a model that only considers restricted classes of adversaries that, in a nutshell, output a group element $z \in \mathbb{G}$ together with its representations $[z]$ w.r.t. all elements they have seen so far. This was to invoke the concrete knowledge soundness analysis of Fiat-Shamir Bulletproofs provided in the AGM [GT21].

In this updated version, we improve upon [GOP+22] by proving simulation-extractability for Fiat-Shamir Bulletproofs in the random oracle model and thus extending it to include all PPT adversaries. The technical core of our analysis relies on the work of [GOP+22] and [AFK21]. Definitions of FS-EXT and FS-SIM-EXT have been changed to more standard ones accordingly. The current version simplifies the generic result showing FS-SIM-EXT assuming FS-WUR and FS-EXT by removing AGM-specific arguments. In particular, the present result does not require the NIZK simulator to be algebraic anymore. The unique response analysis of FS-BP has also been simplified, by directly proving FS-WUR for the Fiat-Shamir

[5] `https://zkproof.org`

[6] The analysis is not directly applicable to Bulletproofs. However, as we will see in Section 2.8, it applies to Bulletproofs after minor tweaks to the protocol.

$$\begin{tikzcd}
\mathsf{k\text{-}SS} \arrow[r, "\text{Theorem 2.1 (Known from [AFK21])}"] & \mathsf{FS\text{-}EXT} \arrow[dr, dash] & & \\
\mathsf{HVZK} \arrow[r, "\text{Programmable ROM}"] \arrow[dr, dash] & \mathsf{NIZK} \arrow[r, dash] & {} \arrow[r, "\text{Lemma 3.2}"] & \mathsf{FS\text{-}SIM\text{-}EXT} \\
\mathsf{SR\text{-}WUR} \arrow[r, "\text{Lemma 3.1}"] & \mathsf{FS\text{-}WUR} \arrow[ur, dash] & &
\end{tikzcd}$$

Fig. 1: Overview of modular security analysis towards simulation-extractability of multi-round Fiat–Shamir NIZK.

transformed protocol instead of its state-restoration variant for the interactive protocol. Along the way, our reduction relies on the extraction strategy of [AFK21].

1.1 Technical Overview

As already argued, in applications where proof systems are deployed, an adversary who tries to break the system has access to proofs provided by other parties using the same scheme. Thus, any reasonable security notion must require that a ZK proof system be secure against adversaries that potentially see and utilise proofs generated by different parties. *Simulation-soundness* and *simulation-extractability* (FS-SIM-EXT) are the notions that guarantee soundness (the prover cannot prove false statements) or the stronger property knowledge-soundness (the prover cannot prove statements without knowing a witness) to hold against adversaries who may see many (simulated) proofs.

From extractability to simulation-extractability. Our starting point is the work of Attema, Fehr and Klooß [AFK21], that proves that the Fiat-Shamir transform of multi-round special-sound (**k**-SS) public coin protocols is knowledge sound (or extractable, denoted by FS-EXT) in the (programmable) random oracle (RO) model. They proposed a novel extractor algorithm that given black-box rewinding access to the adversary extracts a valid witness whenever the adversary outputs an accepting transcript. In particular, their analysis implies that the Fiat-Shamir-transformed *multi-round* public coin protocols, are knowledge sound in the ROM *without an exponential security loss*. This can be extended to Bulletproofs (henceforth referred to as BP), as we will see in Section 2.8. The natural question is then, can their proof be easily extended to cover the case of simulation-extractability (where the result needs to hold even when the NIZK simulator has to provide the adversary with simulated proofs on statements of their choice)?

To see why the extension is not straightforward, consider the following natural approach: From [AFK21] we can conclude (cf. Section 2.8) that Fiat-Shamir BP (FS-BP) satisfies FS-EXT. By proof-by-contradiction, we assume there exists a cheating prover $\hat{\mathcal{P}}$ against FS-SIM-EXT. If we now manage to construct an adversary $\mathcal{P}$ against FS-EXT given $\hat{\mathcal{P}}$, we could conclude that FS-BP is indeed FS-SIM-EXT. A possible way to construct $\mathcal{P}$ invoking $\hat{\mathcal{P}}$ internally is as follows: The proof queries made by $\hat{\mathcal{P}}$ are answered by $\mathcal{P}$ by running the honest verifier zero-knowledge (HVZK) simulator of BP, and then programming the RO with the challenges returned by the simulator. The RO queries, on the other hand, are simply forwarded outside to the RO as before. When $\hat{\mathcal{P}}$ outputs a transcript $\mathcal{T}$, $\mathcal{P}$ forwards the same as its output. However, this reduction fails in certain scenarios, e.g., consider the case when $\mathcal{T}$ shares a common prefix with one of the simulated transcript. The challenges in $\mathcal{T}$ would not correspond to the actual RO output (as they are programmed), and thus, $\mathcal{T}$ would not be an accepting transcript in the knowledge soundness game even if it is accepting in the FS-SIM-EXT game. An existing strategy to circumvent the issue is to assume that no adversary can generate two distinct accepting transcripts that share a common prefix, which is captured by the notion of "unique response" for the underlying protocol. This notion has already been used to prove simulation-extractability of Σ-protocols [FKMV12], multi-round public coin interactive protocols [DFM20, GKK+22], and Sonic and Plonk [GKK+22]. However, BP does not have unique response under their definition: this is simple to see since randomized commitments are sent from the prover during the third round, so one can trivially come up with two accepting transcripts with distinct commitment randomnesses.

The next natural attempt might then be to "de-randomize" later rounds of BP e.g., by letting the prover choose and commit all their random coins in the first round, and then prove consistency of all future rounds with these coins. This of course introduces new challenges, since these additional consistency proofs must themselves not use any additional randomness in rounds other than the first one. While these technical challenges could be overcome using the right tools, the final solution would be all but satisfactory. First of all, the new protocol would be less efficient than the original BP. And perhaps more importantly, all real-world implementations of BP would have to decide whether to switch to the new protocol without any evidence that the original BP is insecure.

Instead, we present a new approach here that allows us to prove that Fiat-Shamir BP *as is* satisfies FS-SIM-EXT, which has wide-reaching impact for systems based on BP that are already in use. We discuss our new security notions and a chain of implications below.

Weak unique response. We introduce two new definitions: state-restoration weak unique response (SR-WUR), and Fiat-Shamir weak unique response (FS-WUR), which are the interactive, and non-interactive definitions for showing unique response of protocols. We show that these two notions are tightly related, i.e., FS-WUR tightly reduces to SR-WUR of the interactive protocol (Lemma 3.1). Both notions require that it should be hard for the adversary, on input a simulated proof, to output a proof which shares a prefix with it. This is opposed to the previous notion of (strong) unique response that requires it should be infeasible for the adversary to come up with two different proofs that share a prefix. As an analogy, our notion is akin to second preimage resistance for hash functions, while the previous notion is akin to collision resistance. Clearly, it is easier to show that an existing protocol satisfies the weaker definition. But it is in turn harder to show that the weaker definition is enough to achieve the overall goal. However, note that the weaker variant of the definition is also somewhat closer to the intuitive goal of non-malleability: we do not want the adversary to be able to reuse parts of proofs generated by other parties to forge new proofs.

Simultation-extractability of multi-round Fiat–Shamir. Once we have FS-EXT, FS-WUR, and NIZK for a non-interactive protocol, we are able to show its simulation-extractability. The diagram in Fig. 1 summarizes our modular security analysis towards FS-SIM-EXT of multi-round Fiat–Shamir NIZK. While our framework has been built with Bulletproofs as its main use case, the results from Section 3 are stated in a self-contained manner and could be used to show simulation-extractability for other public-coin protocols in the literature. Putting our analysis and the result of [ACK21] together, we obtain the following general theorem.

Theorem 1.1 (General Theorem (Informal)). *If a multi-round public-coin interactive protocol satisfies (1) special soundness (**k**-SS), (2) perfect honest verifer zero knowledge (HVZK), and (3) state-restoration weak unique responses (SR-WUR), then the non-interactive version of the protocol achieved via the Fiat-Shamir transform, is simulation-extractable (FS-SIM-EXT) in the programmable random oracle model.*

Non-malleable Bulletproofs. We use our definitional foundation to show that Fiat-Shamir BP is non-malleable and give concrete security bounds for it. The main technical contribution here is to show that FS-BP satisfies our (weaker) definition of unique response, namely FS-WUR.[7] We show that given $\mathcal{A}_{ur}$ breaking FS-WUR, one can construct a reduction $\mathcal{B}$ that finds a non-trivial discrete logarithm relation between group generators in the public parameters, which is already assumed to be hard in [BBB+18] for proving the knowledge soundness of BP. To this end, the reduction $\mathcal{B}$ requires the group representation of forged transcript $\mathcal{T}$. This is enabled by switching to a simulator which remembers the group representation of simulated transcript $\widetilde{\mathcal{T}}$ (which shares some prefix with $\mathcal{T}$) and then by rewinding $\mathcal{A}_{ur}$ to extract the representation of the remaining part. For the latter analysis, we invoke a variant of the extractor algorithm proposed in [AFK21] to construct a sub-tree of transcripts sharing the prefix with $\widetilde{\mathcal{T}}$. For the other assumptions in the theorem, we rely on existing knowledge with some adjustments: BP is known to satisfy FS-EXT (from [AFK21]). We do this for two versions of FS-BP, namely Bulletproofs for arithmetic circuits (in Section 4) and range-proofs Bulletproofs (in Appendix B).

.

1.2 Related work

Goldwasser and Kalai [GK03] show that the Fiat-Shamir heuristic is not sound in general, by showing explicit – and somewhat contrived – counterexamples that cannot be proven secure for any hash function. However, there is no evidence that any *natural* construction using the Fiat-Shamir heuristic is insecure.

[7] While the proceedings version proved SR-WUR of the interactive BP in the AGM, in the present version we show FS-WUR directly in the non-interactive setting. This is to make use of the existing extractor of [ACK21] which was already tailored to the random oracle instead of the state-restoration oracle.

Faust et al. [FKMV12] are the first to analyze simulation-soundness and simulation-extractrability of Fiat–Shamir NIZK from Σ-protocols. Ganesh et al. [GKK+22] extend their result to multi-round protocols with $(n_1, \ldots, n_r)$-special soundness. The latter result requires the Fiat-Shamir transformed protocol to be k-unique response, i.e., it is hard for a PPT adversary to produce two distinct proofs that have the same first k-round messages, and k-programmable trapdoor-less zero-knowledge, i.e., the simulator only needs to program the challenge corresponding to the k-th message. BP simulators, that we know of, need to program challenges starting from round 1. Moreover, as discussed earlier, it has an intermediate randomized round due to which it does not satisfy 1-unique response. Thus, general results from [GKK+22] are not applicable to BP.

Don et al. [DFM20] study multi-round Fiat–Shamir in the quantum random oracle model, but their generic claim (Corollary 15) incurs at least a multiplicative factor $O(q^r)$[8] in the loss in soundness due to Fiat–Shamir, even if the result is downgraded to the classical setting. Hence their result leads to a super-polynomial loss when the number of rounds r depends on the security parameter as in Bulletproofs. They also showed simulation-extractability of multi-round Fiat–Shamir proofs in the QROM assuming the (strong) unique response property of the underlying interactive protocols. As mentioned earlier, Bulletproofs do not meet their definition of unique responses and we are thus motivated to explore alternative paths towards simulation-extractability, but in the classical ROM.

There are a limited number of works that analyze the concrete soundness loss incurred by Fiat–Shamir when applied to *non-constant round* protocols. Ben-Sasson et al. [BCS16] show that if the underlying *interactive oracle proof* protocol satisfies *state-restoration soundness* (a stronger variant of soundness where the prover is allowed to rewind the verifier states) then Fiat–Shamir only introduces $3(q^2+1)2^{-\lambda}$ of additive loss both in soundness and proof of knowledge. Canetti et al. [CCH+18, CCH+19] propose the closely related notion of *round-by-round soundness* which is sufficient to achieve soundness, even in the standard model.Following these works, Holmgren [Hol19] shows state-restoration soundness and round-by-round soundness are equivalent.

Ghoshal and Tessaro [GT21] and Bünz et al. [BMM+19] (concurrently) provide a detailed analysis of *non-interactive* Bulletproofs in the algebraic group model (AGM) [FKL18] and, in particular, the former shows *state-restoration witness extended emulation* of interactive Bulletproofs in the AGM and uses it to argue that extractability of non-interactive Bulletproofs results in $(q+1)/2^{\mathsf{sLen}(\lambda)}$ in additive loss, where $\mathsf{sLen}(\lambda)$ is the bit length of the shortest challenge. Finally, works by [AFK21, Wik21] generically explore the concrete knowledge error of Fiat-Shamir-transformed special-sound multi-round public coin protocols in the ROM, which includes $\mathsf{FS\text{-}BP}$. Crucially, these works present carefully designed extraction strategies tailored to special sound protocols to circumvent the loss of $O(q^r)$ in the knowledge error and expected running time of the extractor. However, none of these works explore simulation-soundness or simulation-extractability of non-constant round Fiat–Shamir.

There are also other zkSNARKs that satisfy simulation-extractability such as e.g., [GM17] and [Gro16, GKK+22]. However, these constructions are very different than Bulletproofs since they rely on a structured reference string which comes with a trapdoor, the knowledge of which compromises the soundness. [BKSV20] show techniques to make [Gro16] black-box weakly simulation-extractable NIZK using verifiable encryption. A generic framework to turn existing zkSNARKs into simulation-extractable zkSNARKs was presented in [ARS20], but Bulletproofs is not covered by their result since their transform only works for schemes with trusted setup.

2 Preliminaries

2.1 Notations

We denote by $\mathbb{N} = \{0, 1, 2, \ldots\}$ the natural numbers, and $\mathbb{Z}$ the integers. We use bold face to denote vectors, e.g., $\mathbf{a} \in \mathbb{Z}^n$ is a vector of n integers. For $\mathbf{a}, \mathbf{b} \in \mathbb{Z}^n$ we write $\langle \mathbf{a}, \mathbf{b} \rangle$ for the inner product. For a group $\mathbb{G}$ where $(g_1, \ldots, g_n) = \mathbf{g} \in \mathbb{G}^n$ and $(a_1, \ldots, a_n) = \mathbf{a} \in \mathbb{Z}^n$, we denote by $\mathbf{g}^{\mathbf{a}} = \prod_{i=1}^{n} g_i^{a_i}$. The identity element of $\mathbb{G}$ is denoted by $1_{\mathbb{G}}$. The Hadamard product of vectors $\mathbf{a} = (a_1, \ldots, a_n)$ and $\mathbf{b} = (b_1, \ldots, b_n)$ is $\mathbf{a} \circ \mathbf{b} = (a_1 b_1, \ldots, a_n b_n)$. For $z \in \mathbb{Z}_p^*$, denote $\mathbf{z}^n$ as the vector $(1, z, \ldots, z^{n-1})$, and $\mathbf{z}^{-n} = (1, z^{-1}, \ldots, z^{-n+1})$.

The security parameter λ is 1^λ in unary. We use $\mathsf{negl}(\lambda)$ to denote a negligible function in λ. We write $X_\lambda \equiv Y_\lambda$ to denote that probabilistic ensembles $\{X_\lambda\}_\lambda$ and $\{Y_\lambda\}_\lambda$ are equivalent.

[8] Here and below q is the number of queries to the random oracle, r is the number of rounds, and λ is the security parameter.

2.2 Discrete Logarithm Problem

We recall the discrete logarithm relation problem (DL-REL). It is defined with respect to a group generator algorithm $\mathsf{GGen}(1^\lambda)$ that outputs a cyclic group $\mathbb{G}$ of prime order $p(\lambda)$. Note that DL-REL is asymptotically equivalent to the standard discrete logarithm assumption (see e.g. [JT20, Lemma 3]).

Definition 2.1 (DL-REL). *Given an adversary $\mathcal{A}$ and $n \geq 2$, the advantage of finding a non-trivial discrete logarithm relation between random n generators is*

$$\mathbf{Adv}_{\mathsf{GGen}}^{\mathit{DL\text{-}REL}}(\mathcal{A}) = \Pr\left[\mathbf{g}^{\mathbf{x}} = 1_{\mathbb{G}} \wedge \mathbf{x} \neq \mathbf{0} \ : \ \begin{array}{r} \mathbb{G} \leftarrow \mathsf{GGen}(1^\lambda); \mathbf{g} := (g_1, \ldots, g_n) \xleftarrow{\$} \mathbb{G}^n; \\ \mathbf{x} := (x_1, \ldots, x_n) \leftarrow \mathcal{A}(\mathbb{G}, \mathbf{g}) \end{array}\right].$$

2.3 Relations

Let $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^*$ be an efficiently decidable relation. For given public parameters pp, we call w a *witness* for *statement* x if $(\mathsf{pp}, x, w) \in \mathcal{R}$. The associated language for pp is $\mathcal{L}_{\mathsf{pp}} = \{x \mid \exists w \text{ s.t. } (\mathsf{pp}, x, w) \in \mathcal{R}\}$. Given a fixed public parameter pp, we denote by $\mathcal{R}_{\mathsf{pp}}$ the restriction of $\mathcal{R}$ to $\{(x, w) \ : \ (\mathsf{pp}, x, w) \in \mathcal{R}\}$.

Relation for inner product argument. The relation proven by the inner-product argument in [BBB+18] is

$$\begin{aligned} \mathcal{R}_{\mathsf{InPrd}} &= \{((n, \mathbf{g}, \mathbf{h}, u), (P), (\mathbf{a}, \mathbf{b})) \mid P = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}u^c \wedge c = \langle \mathbf{a}, \mathbf{b}\rangle\} \\ &\subseteq (\mathbb{N} \times \mathbb{G}^n \times \mathbb{G}^n \times \mathbb{G}) \times (\mathbb{G}) \times (\mathbb{Z}_p^n \times \mathbb{Z}_p^n) \end{aligned} \tag{1}$$

where $\mathbb{G} \in \mathsf{GGen}(1^\lambda)$ and $\mathbf{g}, \mathbf{h}$ are vectors of uniformly sampled independent generators.

Relation for arithmetic circuit satisfiability. Any arithmetic circuit with n multiplication gates can be represented using a constraint system (cf. [BCC+16]). The constraint system consists of three vectors $\mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O \in \mathbb{Z}_p^n$ representing the left inputs, right inputs, and outputs of multiplication gates respectively, so that $\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O$, and additional $Q \leq 2n$ linear constraints. The linear constraints can be represented as $\mathbf{W}_L \cdot \mathbf{a}_L + \mathbf{W}_R \cdot \mathbf{a}_R + \mathbf{W}_O \cdot \mathbf{a}_O = \mathbf{c}$, where $\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O \in \mathbb{Z}_p^{Q \times n}$ and $\mathbf{c} \in \mathbb{Z}_p^Q$. This results in the following relation for arithmetic circuit satisfiability

$$\begin{aligned} \mathcal{R}_{\mathsf{ACS}} &= \left\{ \begin{array}{l} ((n, Q), (\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c}), (\mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O)) \mid \\ \mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O \wedge \mathbf{W}_L \cdot \mathbf{a}_L + \mathbf{W}_R \cdot \mathbf{a}_R + \mathbf{W}_O \cdot \mathbf{a}_O = \mathbf{c} \end{array} \right\} \\ &\subseteq (\mathbb{Z} \times \mathbb{Z}) \times (\mathbb{Z}_p^{Q \times n} \times \mathbb{Z}_p^{Q \times n} \times \mathbb{Z}_p^{Q \times n} \times \mathbb{Z}_p^Q) \times (\mathbb{Z}_p^n \times \mathbb{Z}_p^n \times \mathbb{Z}_p^n) \end{aligned}$$

For simplicity, we also include the Bulletproofs setup resulting in the relation

$$\begin{aligned} \mathcal{R}_{\mathsf{ACSPf}} &= \left\{ \begin{array}{l} ((n, Q, g, h, u, \mathbf{g}, \mathbf{h}), (\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c}), (\mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O)) \mid \\ \mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O \wedge \mathbf{W}_L \cdot \mathbf{a}_L + \mathbf{W}_R \cdot \mathbf{a}_R + \mathbf{W}_O \cdot \mathbf{a}_O = \mathbf{c} \end{array} \right\} \\ &\subseteq (\mathbb{Z}^2 \times \mathbb{G}^3 \times \mathbb{G}^n \times \mathbb{G}^n) \times (\mathbb{Z}_p^{Q \times n} \times \mathbb{Z}_p^{Q \times n} \times \mathbb{Z}_p^{Q \times n} \times \mathbb{Z}_p^Q) \times (\mathbb{Z}_p^n \times \mathbb{Z}_p^n \times \mathbb{Z}_p^n) \end{aligned} \tag{2}$$

2.4 Interactive Proofs

An *interactive proof* [GMR85] is a tuple of three algorithms $\mathit{\Pi} = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$. On input 1^λ, the setup algorithm Setup produces public parameters pp. The *transcript* of the interaction between prover $\mathcal{P}$ with private input w (i.e., *witness*) and PPT verifier $\mathcal{V}$ with common input x (i.e., *statement*) is denoted by $\mathcal{T} \leftarrow \langle \mathcal{P}(\mathsf{pp}, x, w), \mathcal{V}(\mathsf{pp}, x)\rangle$. After interaction, $\mathcal{V}$ outputs a decision bit b, i.e., $b = 1$ means the verifier accepts. For convenience, Ver denotes the *verdict algorithm* that $\mathcal{V}$ uses to determine the decision bit b at the end of interaction, i.e., $\mathcal{V}$ outputs 1 during $\mathcal{T} \leftarrow \langle \mathcal{P}(\mathsf{pp}, x, w), \mathcal{V}(\mathsf{pp}, x)\rangle$ if and only if $\mathsf{Ver}(\mathsf{pp}, x, \mathcal{T})$ outputs 1. An interactive proof where the prover is PPT is also known as *argument*.

Intuitively an interactive proof for relation $\mathcal{R}$ allows a prover knowing a witness w to convince the verifier of a statement x. Given $(\mathsf{pp}, x, w) \in \mathcal{R}$ an honest prover should always convince the verifier (completeness) while a prover should not be able to convince the verifier of x if x is not true (soundness), nor without 'knowing' w such that $(\mathsf{pp}, x, w) \in \mathcal{R}$ (proof of knowledge). Finally, the proof should not reveal any more information than the validity of the statement (zero-knowledge).

In this work, we focus on analyzing $(2r+1)$-*round proof systems* where the prover (resp. the verifier) sends messages in odd (resp. even) rounds. That is, a transcript $\mathcal{T}$ of $(2r+1)$-round proof can be parsed as $\mathcal{T} = (a_1, c_1, \ldots, a_r, c_r, a_{r+1})$ where a_i for $i = 1, \ldots, r+1$ is a $(2i-1)$-th round message output by $\mathcal{P}$ and c_i for $i = 1, \ldots, r$ is a $2i$-th round message output by $\mathcal{V}$, respectively.

Definition 2.2 (Public Coin). *An interactive proof $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ is* public coin *if all messages sent by the verifier $\mathcal{V}$ during the interaction with $\mathcal{P}$ in round $2i$ are chosen uniform-randomly from the challenge space Ch_i (where the challenge spaces Ch_i are specified by Setup). In particular, they are independent of the provers messages.*

Definition 2.3 (Completeness). *An interactive proof $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ satisfies* perfect completeness *if for all λ and for all adversaries $\mathcal{A}$ it holds that*

$$\Pr\left[\mathsf{Ver}(\mathsf{pp}, x, \mathcal{T}) = 1 \ \wedge \ (x, w) \in \mathcal{R}_{\mathsf{pp}} \ : \ \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); \\ (x, w) \leftarrow \mathcal{A}(\mathsf{pp}) \\ \mathcal{T} \leftarrow \langle \mathcal{P}(\mathsf{pp}, x, w)\,,\, \mathcal{V}(\mathsf{pp}, x)\rangle \end{array}\right] = 1.$$

Definition 2.4 (Perfect Honest-Verifier Zero-Knowledge). *An interactive proof $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ for relation $\mathcal{R}$ is perfect* honest verifier zero-knowledge *(HVZK) if there exists a PPT simulator $\mathcal{S}$ such that for all adversaries $\mathcal{A}$ it holds that*

$$\langle \mathcal{P}(\mathsf{pp}, x, w)\,,\, \mathcal{V}(\mathsf{pp}, x)\rangle \equiv \mathcal{S}(\mathsf{pp}, x)$$

where $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and $(x, w) \leftarrow \mathcal{A}(\mathsf{pp})$ such that $(x, w) \in \mathcal{R}_{\mathsf{pp}}$.

Min-entropy of commitments. We want to assess how likely it is for first round messages of an interactive protocol to collide with a fixed value.

Definition 2.5 (Min-entropy of commitments). *Let $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ be a multi-round interactive proof for relation $\mathcal{R}$. Let $\mathsf{Coin}(\lambda)$ be the set of random coins used by the prover on input $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$ and $\mathcal{P}_1(\cdot)$ the function that outputs the first round message for $\mathcal{P}$. For any fixed $\lambda \in \mathbb{N}$ and $(\mathsf{pp}, x, w) \in \mathcal{R}$ such that $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$, consider the maximum probability that a first round prover message hits a particular value:*

$$\mu(\lambda, \mathsf{pp}, x, w) = \max_{a_1 \in \{0,1\}^*} \Pr\left[\mathcal{P}_1(\mathsf{pp}, x, w; \rho) = a_1 \ : \ \rho \xleftarrow{\$} \mathsf{Coin}(\lambda)\right].$$

The min-entropy *α of protocol Π is*

$$\alpha(\lambda) = \min_{\mathsf{pp} \in \mathsf{Setup}(1^\lambda) \ \wedge \ (x,w) \in \mathcal{R}_{\mathsf{pp}}} (-\log_2(\mu(\lambda, \mathsf{pp}, x, w))).$$

2.5 Non-Interactive Proofs via Fiat–Shamir

In this paper we focus on *non-interactive proof systems* obtained via the Fiat–Shamir heuristic [FS87], which allows to transform a public-coin interactive proof into an non-interactive version in the random oracle model. Given an interactive proof system $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$, we define the corresponding non-interactive proof system $\Pi_{\mathsf{FS}} = (\mathsf{Setup}_{\mathsf{FS}}, \mathcal{P}_{\mathsf{FS}}, \mathcal{V}_{\mathsf{FS}})$ as follows. We write $\mathcal{P}^{\mathsf{H}}_{\mathsf{FS}}$ and $\mathcal{V}^{\mathsf{H}}_{\mathsf{FS}}$ to denote that the prover and verifier only have oracle access to H.

– We denote by $(\mathsf{pp}, \mathsf{H}) \leftarrow \mathsf{Setup}_{\mathsf{FS}}(1^\lambda)$ the setup algorithm generating pp by invoking $\mathsf{Setup}(1^\lambda)$ of Π, and then for $i \in [1, r]$ samples a function H_i uniformly from a set of all functions that map $\{0,1\}^*$ to Ch_i. Note that this is equivalent to instantiating H_i from a single random oracle via usual domain separation. We denote H as shorthand for $\{\mathsf{H}_i\}_{i \in [1,r]}$.

– We denote by $\mathcal{T} \leftarrow \mathcal{P}^{\mathsf{H}}_{\mathsf{FS}}(\mathsf{pp}, x, w)$ the prover producing a proof string $\mathcal{T}$[9] on input pp, statement x, and witness w. For each round $i \in [1, r]$, $\mathcal{P}^{\mathsf{H}}_{\mathsf{FS}}$ invokes the next message function of the interactive prover $\mathcal{P}(\mathsf{pp}, x, w)$ on prior challenge c_{i-1} to get a_i, and obtains the ith round challenge by computing $c_i = \mathsf{H}_i(\mathsf{pp}, x, a_1, c_1, \ldots, a_{i-1}, c_{i-1}, a_i)$. Then $\mathcal{P}^{\mathsf{H}}_{\mathsf{FS}}$ outputs $\mathcal{T} = (a_1, c_1, \ldots, a_r, c_r, a_{r+1})$.

– We write $b \leftarrow \mathcal{V}^{\mathsf{H}}_{\mathsf{FS}}(\mathsf{pp}, x, \mathcal{T})$ for the decision bit of the verifier on input of pp, statement x, and proof string $\mathcal{T}$. $\mathcal{V}^{\mathsf{H}}_{\mathsf{FS}}$ outputs $b = 1$, meaning the verifier accepts the proof, iff $\mathsf{Ver}(\mathsf{pp}, x, \mathcal{T}) = 1$ *and* $c_i = \mathsf{H}_i(\mathsf{pp}, x, a_1, c_1, \ldots, c_{i-1}, a_i)$ for all $i \in [1, r]$.

Definition 2.6 (Completeness). *A non-interactive proof $\Pi_{\mathsf{FS}} = (\mathsf{Setup}_{\mathsf{FS}}, \mathcal{P}_{\mathsf{FS}}, \mathcal{V}_{\mathsf{FS}})$ for relation $\mathcal{R}$ satisfies* perfect completeness *if for all λ and for all adversaries $\mathcal{A}$ it holds that*

$$\Pr\left[\mathcal{V}^{\mathsf{H}}_{\mathsf{FS}}(\mathsf{pp}, x, \mathcal{T}) = 1 \wedge (x, w) \in \mathcal{R}_{\mathsf{pp}}; \ : \ \begin{array}{r} (\mathsf{pp}, \mathsf{H}) \leftarrow \mathsf{Setup}_{\mathsf{FS}}(1^\lambda); \\ (x, w) \leftarrow \mathcal{A}(\mathsf{pp}) \\ \mathcal{T} \leftarrow \mathcal{P}^{\mathsf{H}}_{\mathsf{FS}}(\mathsf{pp}, x, w) \end{array}\right] = 1.$$

[9] To make our analysis simpler we include a version of Fiat-Shamir where a proof string contains all challenges. This also allows us to use the same notation $\mathcal{T}$ for transcripts and proof strings.

2.6 NIZK and Simulation Oracles

We define zero-knowledge for non-interactive proofs in the explicitly programmable random oracle model where the simulator can program the random oracle. The formalization below can be seen as that of [FKMV12] adapted to multi-round protocols. The zero-knowledge simulator $\mathcal{S}_{\mathsf{FS}}$ is defined as a stateful algorithm that operates in two modes. In the first mode, $(c_i, \mathsf{st}') \leftarrow \mathcal{S}_{\mathsf{FS}}(1, \mathsf{st}, t, i)$ takes care of random oracle calls to H_i on input t. In the second mode, $(\widetilde{\mathcal{T}}, \mathsf{st}') \leftarrow \mathcal{S}_{\mathsf{FS}}(2, \mathsf{st}, x)$ simulates a valid proof string. Throughout the paper, we assume the state st is initialized with pp once it's generated by $\mathsf{Setup}_{\mathsf{FS}}$. For convenience we define three "wrapper" oracles. These oracles are stateful and share state.

- $\mathcal{S}_1(t, i)$ to denote the oracle that returns the first output of $\mathcal{S}_{\mathsf{FS}}(1, \mathsf{st}, t, i)$;
- $\mathcal{S}_2(x, w)$ that returns the first output of $\mathcal{S}_{\mathsf{FS}}(2, \mathsf{st}, x)$ if $(\mathsf{pp}, x, w) \in \mathcal{R}$ and $\perp$ otherwise;
- $\mathcal{S}'_2(x)$ that returns the first output of $\mathcal{S}_{\mathsf{FS}}(2, \mathsf{st}, x)$.

Since NIZK is a security property that is only guaranteed for valid statements in the language, the definition below makes use of $\mathcal{S}_2$ as a proof simulation oracle. As we shall see later, *simulation-extractability* on the other hand is defined with respect to an oracle similar to $\mathcal{S}'_2$ following [FKMV12].

Definition 2.7 (Non-interactive Zero Knowledge). *Let $\Pi_{\mathsf{FS}} = (\mathsf{Setup}_{\mathsf{FS}}, \mathcal{P}^{\mathsf{H}}_{\mathsf{FS}}, \mathcal{V}^{\mathsf{H}}_{\mathsf{FS}})$ be a non-interactive proof for relation $\mathcal{R}$. Π_{FS} is unbounded* non-interactive zero knowledge *(NIZK) in the random oracle model, if there exist a PPT simulator $\mathcal{S}_{\mathsf{FS}}$ with wrapper oracles $\mathcal{S}_1$ and $\mathcal{S}_2$ such that for all PPT distinguisher $\mathcal{D}$, its advantage*

$$\begin{aligned}&\mathbf{Adv}^{\mathsf{NIZK}}_{\Pi_{\mathsf{FS}}}(\mathcal{D}, \mathcal{S}_{\mathsf{FS}})\\ =&\left|\Pr\left[\mathcal{D}^{\mathsf{H}, \mathcal{P}^{\mathsf{H}}_{\mathsf{FS}}}(\mathsf{pp}) = 1 \ : \ (\mathsf{pp}, \mathsf{H}) \leftarrow \mathsf{Setup}_{\mathsf{FS}}(1^\lambda)\right] - \Pr\left[\mathcal{D}^{\mathcal{S}_1, \mathcal{S}_2}(\mathsf{pp}) = 1 \ : \ (\mathsf{pp}, \mathsf{H}) \leftarrow \mathsf{Setup}_{\mathsf{FS}}(1^\lambda)\right]\right|\end{aligned}$$

is negligible in λ, where both $\mathcal{P}^{\mathsf{H}}_{\mathsf{FS}}(\mathsf{pp}, x, w)$ and $\mathcal{S}_2$ return $\perp$ if $(\mathsf{pp}, x, w) \notin \mathcal{R}$.

Given a perfect HVZK simulator $\mathcal{S}$ for Π, we immediately obtain the following *canonical* NIZK simulator $\mathcal{S}_{\mathsf{FS}}$ for Π_{FS} by defining responses of each mode as follows.

- To answer query (t, i) with mode 1, $\mathcal{S}_{\mathsf{FS}}(1, \mathsf{st}, t, i)$ lazily samples a lookup table $\mathcal{Q}_{1,i}$ kept in state st. It checks whether $\mathcal{Q}_{1,i}[t]$ is already defined. If this is the case, it returns the previously assigned value; otherwise it returns and sets a fresh random value c_i sampled from Ch_i.
- To answer query x with mode 2, $\mathcal{S}_{\mathsf{FS}}(2, \mathsf{st}, x)$ calls the perfect HVZK simulator $\mathcal{S}$ of Π to obtain a simulated transcript $\mathcal{T} = (a_1, c_1, \ldots, a_r, c_r, a_{r+1})$. Then, it programs the tables such that $\mathcal{Q}_{1,1}[\mathsf{pp}, x, a_1] := c_1, \ldots, \mathcal{Q}_{1,r}[\mathsf{pp}, x, a_1, c_1, \ldots, a_r] := c_r$. If any of the table entries has been already defined $\mathcal{S}_{\mathsf{FS}}$ aborts, which should happen with negligible probability assuming high min-entropy of a_1.

2.7 Knowledge Extraction in ROM

We define an adaptive version of extractability for Fiat-Shamir NIZK. We slightly modify Definition 10 of [AFK21] by explicitly passing the random oracle query history of the first run to an extractor, and by having an extractor only output the witness. Since the extractor of [AFK21] initially invokes a prover and simulates the random oracle responses honestly, their main theorem can be reused to prove FS-EXT.[10]

Definition 2.8 (FS-EXT security). *Consider a non-interactive proof system $\Pi_{\mathsf{FS}} = (\mathsf{Setup}_{\mathsf{FS}}, \mathcal{P}_{\mathsf{FS}}, \mathcal{V}_{\mathsf{FS}})$ for relation $\mathcal{R}$. Π_{FS} is (adaptively)* extractable *(FS-EXT) with knowledge error $\kappa : \mathbb{N} \times \mathbb{N} \rightarrow [0, 1]$ in the random oracle model, if there exists an extractor $\mathcal{E}$ and some polynomial* poly*, such that for any* PPT *adversary $\mathcal{P}$ that makes at most q queries to H, it holds that*

$$\mathrm{ext}(\mathcal{P}, \mathcal{E}) \geq \frac{\mathrm{acc}(\mathcal{P}) - \kappa(\lambda, q)}{\mathsf{poly}(\lambda)}$$

and $\mathcal{E}$ halts in an expected number of steps that is polynomial in λ and q, where the probabilities acc *(taken over the random coins used by* $\mathsf{Setup}_{\mathsf{FS}}$ *and $\mathcal{P}$) and* ext *(taken over the random coins used by* $\mathsf{Setup}_{\mathsf{FS}}$*, $\mathcal{P}$,*

[10] We also omit auxiliary information of [AFK21, Definition 10] since in our formulation the distributions of outputs of $\mathcal{P}$ are trivially identical in acc and ext.

and $\mathcal{E}$) are defined as follows.

$$\mathrm{acc}(\mathcal{P}) = \Pr\left[b = 1 \;:\; (\mathsf{pp}, \mathsf{H}) \leftarrow \mathsf{Setup}_{\mathsf{FS}}(1^\lambda); (x, \mathcal{T}) \leftarrow \mathcal{P}^{\mathsf{H}}(\mathsf{pp}; \rho); b \leftarrow \mathcal{V}_{\mathsf{FS}}^{\mathsf{H}}(\mathsf{pp}, x, \mathcal{T})\right]$$

$$\mathrm{ext}(\mathcal{P}, \mathcal{E}) = \Pr\left[b = 1 \;\wedge\; (x, w) \in \mathcal{R}_{\mathsf{pp}} \;:\; \begin{array}{c}(\mathsf{pp}, \mathsf{H}) \leftarrow \mathsf{Setup}_{\mathsf{FS}}(1^\lambda); (x, \mathcal{T}) \leftarrow \mathcal{P}^{\mathsf{H}}(\mathsf{pp}; \rho);\\ b \leftarrow \mathcal{V}_{\mathsf{FS}}^{\mathsf{H}}(\mathsf{pp}, x, \mathcal{T}); w \leftarrow \mathcal{E}^{\mathcal{P}}(\mathsf{pp}, x, \mathcal{T}, \rho, \mathcal{Q}_1)\end{array}\right]$$

where $\mathcal{Q}_1 = \{\mathcal{Q}_{1,i}\}_{i\in[1,r]}$ is a set of query response pairs corresponding to queries to the random oracle H *with index i. In the experiment of* ext, *$\mathcal{E}$ has oracle access to the next-message function of $\mathcal{P}$.*

2.8 Knowledge Extraction for Special-Sound Multi-Round Protocols

Attema, Fehr and Klooß [AFK21] presented a proof that multi-round public coin protocols with generalized special soundness (defined below) lead to extractable non-interactive proofs in the random oracle model when the Fiat-Shamir transform is applied. Importantly, they carefully design a knowledge extractor that significantly reduces number of rewindings via a clever *early abort* strategy. We will use their extractor for proving the concrete result for BP. We summarize the extractor algorithm from [AFK21] in the adaptive setting and its implication for (computational) knowledge soundness of BP next.

Definition 2.9 (Tree of Transcripts [AFK21]). *Let $\mathbf{k} = (k_1, \ldots, k_r) \in \mathbb{N}^r$. A* **k***-tree of transcripts for a $(2r + 1)$-round public-coin interactive proof $\mathit{\Pi} = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ is a set of $K = \prod_{i=1}^{r} k_i$ transcripts arranged in the following tree structure. The nodes in this tree correspond to the prover's messages and the edges to the verifier's challenges. Every node at depth i has precisely k_i children corresponding to k_i pairwise distinct challenges. Every transcript corresponds to exactly one path from the root node to a leaf node.*

Definition 2.10 (k-Special-Soundness [AFK21]). *Let $\mathbf{k} = (k_1, \ldots, k_r) \in \mathbb{N}^r$. A $(2r + 1)$-round public-coin interactive proof $\mathit{\Pi}$ for relation $\mathcal{R}$ is* **k**-special-sound *(***k***-SS) if there exists a polynomial time algorithm that, on input a statement x and a* **k***-tree of accepting transcripts outputs a witness w such that $(\mathsf{pp}, x, w) \in \mathcal{R}$.*

The knowledge extractor of [AFK21].

Let $\mathit{\Pi}_{\mathsf{FS}}$ be the Fiat-Shamir transformation of a $\mathbf{k} = (k_1, \ldots, k_r)$-special sound, $(2r + 1)$ rounds public coin interactive proof $\mathit{\Pi}$. Let $\mathcal{P}$ be the adversary against knowledge-soundness of $\mathit{\Pi}_{\mathsf{FS}}$. After making q queries to the random oracle H, $\mathcal{P}$ outputs a transcript $\mathcal{T} = (a_1, c_1 \ldots, a_{r+1})$ together with a statement x. For ease of understanding we denote the following according to the notation in [AFK21]; $I_1 = (x, a_1), I_2 = (x, a_1, a_2), \ldots, I_r = (x, a_1, \ldots, a_r)$. $\mathbf{I} = (I_1, \ldots, I_r)$, and $b = \mathcal{V}^{\mathsf{H}}(\mathsf{pp}, x, \mathcal{T})$. Given $\mathcal{P}$ that outputs $(x, \mathcal{T})$ after querying H, one can construct a wrapper $\mathcal{A}$ that internally invokes $\mathcal{P}$ and outputs $(\mathbf{I}, \mathcal{T}, b)$.

In order to extract, the goal is to construct a **k**-tree given blackbox access to the corrupt prover $\mathcal{A}$. Given such a tree the extractor either extracts a valid witness (because of special-soundness, see Definition 2.10) or breaks discrete-log with very high probability. A **k**-tree is nothing but a composition of k_1 appropriate $(1, k_2, \ldots, k_r)$-trees. Following this intuition, [AFK21] defines extraction algorithm $\mathcal{E}_1$ by defining a sequence of sub-extractors $\mathcal{E}_2, \ldots, \mathcal{E}_r$, where extractor $\mathcal{E}_\ell$ is responsible for constructing a $(1, 1, \ldots, k_\ell, \ldots, k_r)$-tree. The overall extraction algorithm then is recursive calls to these sub-extractors starting from $\mathcal{E}_1$. The recursive definition for $\mathcal{E}_\ell$ appears in Fig. 1. In this definition, calls to $\mathcal{E}_{\ell+1}$ are in fact calls to the adversarial algorithm $\mathcal{A}$, and we assume that challenges for all rounds come from the same challenge set Ch.

Each extractor $\mathcal{E}_1, \ldots, \mathcal{E}_r$ executes in two parts. In the first part, $\mathcal{E}_\ell$ calls $\mathcal{E}_{\ell+1}$ recursively till $\mathcal{A}$ is invoked for the first time. $\mathcal{E}_\ell$ forwards any RO queries made by $\mathcal{E}_{\ell+1}$ outside to the calling sub-extractor $\mathcal{E}_{\ell-1}$. These RO queries will ultimately be forwarded to $\mathcal{E}_1$ who answers them by lazily sampling from the challenge space. The base of the recursion is the algorithm $\mathcal{A}$ which outputs $\mathbf{I}$ after making q RO queries to $\mathcal{E}_r$. This ends the first recursion. Now, in the second part (the repeat loop, see Fig. 1), each sub-extractor $\mathcal{E}_\ell$ calls $\mathcal{E}_{\ell+1}$ to build a $(1, \ldots, k_\ell, \ldots, k_r)$-tree with respect to $\mathbf{I}$, i.e., for all the transcripts in the tree the first $a_1, \ldots, a_\ell$ prover messages are the same as in $\mathbf{I}$. In this part, $\mathcal{E}_\ell$ answers the RO queries made by $\mathcal{E}_{\ell+1}$ itself. It will reprogram the RO output for the ℓ-th round challenge, and otherwise, it answers them consistent with all previous runs of $\mathcal{E}_{\ell+1}$ so far. $\mathcal{E}_\ell$ calls $\mathcal{E}_{\ell+1}$ repeatedly till either an appropriate tree (with respect to $\mathbf{I}$) is found or it runs out of all challenges.

The success probability of extractor $\mathcal{E}_\ell$, i.e. the extractor outputs a valid tree is given as:

Lemma 2.1 (Proposition 2 in [AFK21]). *Let $K_\ell = \prod_{i=\ell}^{r} k_i$. The extractor $\mathcal{E}_\ell$ makes an expected number of at most $K_\ell + q \cdot (K_\ell - 1)$ queries to $\mathcal{A}$ (and thus to $\mathcal{P}$) and successfully outputs $b = 1$ with probability at least*

$$\frac{\epsilon(\mathcal{A}) - (q+1) \cdot \kappa_\ell}{1 - \kappa_\ell}$$

where $\kappa_\ell = Er((k_\ell, \dots, k_r); \mathsf{Ch}_\ell, \dots, \mathsf{Ch}_r)$ is the knowledge error of the interactive proof Π, and $\epsilon(\mathcal{A})$ is the probability that the adversary $\mathcal{A}$ outputs an accepting transcript after making q queries to the random oracle.

We assume that the challenges for all the rounds come from the same set Ch. The term $Er((k_\ell, \dots, k_r); \mathsf{Ch})$ denotes the knowledge error and is recursively defined as:

$$\begin{aligned} Er(\varnothing; \varnothing) &= 1, \text{ and} \\ Er((k_\ell, \dots, k_r); \mathsf{Ch}) &= 1 - \frac{|\mathsf{Ch}| - k_\ell + 1}{|\mathsf{Ch}|}(1 - Er((k_{\ell+1}, \dots, k_r); \mathsf{Ch})) \\ &= 1 - \prod_{i=\ell}^{m} \frac{|\mathsf{Ch}| - k_i + 1}{|\mathsf{Ch}|} \end{aligned}$$

The extractability of Fiat-Shamir-transformed multi-round public coin interactive protocols in the adaptive setting is given by Theorem 2.1.

Theorem 2.1 (Adapted from Theorem 4 in [AFK21]). *The Fiat-Shamir transformation Π_{FS} of a $\mathbf{k} = (k_1, \dots, k_r)$-special-sound interactive proof Π, in which all challenges are sampled from a set Ch of size $|\mathsf{Ch}|$, is FS-EXT with knowledge error*

$$\kappa(\lambda, q) = (q+1) \cdot \kappa_\Pi(\lambda)$$

where $\kappa_\Pi(\lambda) = Er((k_1, \dots, k_r); \mathsf{Ch})$ is the knowledge error of the interactive proof Π.

Remark 2.1. We note that Proposition 2 and Theorem 4 in [AFK21] are stated w.r.t. deterministic and unbounded cheating provers. These extend to PPT provers as well since we can invoke the subextractors to obtain a tree of accepting transcripts even from an argument (computationally bounded) prover with a fixed random tape. Extraction of a witness from such a tree always succeeds in the case of a proof system whereas in an argument, it succeeds with all but negligible probability. This is because one can think of the argument system as a proof of knowledge for a slightly different relation: knowledge of witness for the underlying relation OR a discrete log relation (in general, solution to some computational problem). Thus, given a tree, we either extract the orginal witness or the new witness (break a computational assumption).[11]

Applicability to Bulletproofs [BBB⁺18]. Let n be the dimension of witness vectors in relation $\mathcal{R}_{\mathsf{ACSPf}}$ and $m := \log(n)$. The interactive Bulletproofs protocol for $\mathcal{R}_{\mathsf{ACSPf}}$ (detailed in Protocol 1) is a $(2r+1)$-round public-coin proof with $r = 3 + m$. We first remark a small gap between the assumption required by Theorem 2.1 and Bulletproofs. In fact, the first round challenge of Bulletproofs consists of two values $c_1 = (y, z) \in \mathbb{Z}_p^2$ and for the witness extraction to succeed one needs a sufficient number of *distinct y's and z's individually*, instead of distinct *pairs* (y, z). Concretely, the knowledge soundness analysis of BP in [BBB⁺18] requires n distinct y and $Q+1$ distinct z, respectively. To satisfy the aforementioned $\mathbf{k}$-special soundness the parameter k_1 must be set to $\Omega(p)$, which is exponential in λ and thus invalidates the analysis of [AFK21]. To circumvent the issue, one can slightly tweak the underlying multi-round protocol by introducing an additional dummy round: on receiving a_1 from the prover, the verifier only returns the fist part of the challenge $c_{1,0} = y$, the prover sends an empty string to the verifier, and then the verifier returns the second part $c_{1,1} = z$. This is indeed a $(k_{1,0}, k_{1,1}, k_2, \dots, k_r)$-special sound protocol with $(k_{1,0}, k_{1,1}, k_2, k_3, k_4, \dots, k_r) = (n, Q+1, 7, 2, 8 \dots, 8)$, and therefore applying the Fiat-Shamir, $\mathcal{P}_{\mathsf{FS}}$ derives $y = \mathsf{H}_1(\mathsf{pp}, x, a_1, 0)$ and $z = \mathsf{H}_1(\mathsf{pp}, x, a_1, y, 1)$, respectively. We can thus plug in this protocol into Theorem 2.1. We also remark that the extractor of [AFK21] halts in *expected* polynomial time. Since a reduction solving the DL-REL problem runs this extractor internally, the security of FS-BP also relies on the expected-time hardness of DL-REL, which is shown to hold in the generic group model [JT20].

[11] We thank Thomas Attema for clarifying this extension.

Algorithm 1: $\mathcal{E}_\ell$

```
1. First Run: Run $\mathcal{E}_{\ell+1}$ as follows to obtain $(\mathbf{I}, \mathsf{tr}_1, b)$. Relay all the random oracle queries externally
   to the random oracle and record query response pair. Set $j := I_\ell$ and let $c_j$ be the response
   to query $j$.
2. If $b = 0$, abort with output $b := 0$.
3. Repeat Run: Else, repeat:
   (a) Sample $c' \in \mathsf{Ch} \setminus \{c_j\}$ (without replacement);
   (b) Run $\mathcal{E}_{\ell+1}$ as follows to obtain $(\mathbf{I}', \mathsf{tr}', b')$, aborting right after the initial run of $\mathcal{A}$ if $I'_\ell \neq I_\ell$:
       answer the query to $j$ with $c'_j$, while answering all other queries consistently if the query
       was performed by $\mathcal{E}_{\ell+1}$ already on a previous run and with a fresh random value in $\mathsf{Ch}$
       otherwise;
   until either $k_\ell - 1$ additional challenges $c'_j$ with $b' = 1$ and $I'_\ell = I_\ell$ have been found or until
   all challenges $c' \in \mathsf{Ch}$ have been tried.
4. In the former case, output $\mathbf{I}$, the $k_\ell$ accepting $(1, \ldots, 1, k_{\ell+1}, \ldots, k_r)$-trees $\mathsf{tr}_1, \ldots, \mathsf{tr}_{k_\ell}$, and
   $b := 1$; in the latter case, output $b := 0$.
```

Corollary 2.1 (Knowledge Soundness of FS-BP). *Assuming the expected time hardness of solving the discrete logarithm relation problem, the Fiat-Shamir transformation of Bulletproofs (Protocol 1) is FS-EXT with negligible knowledge error. Concretely, there exists an adversary $\mathcal{B}$ against DL-REL such that Π_{FS} is FS-EXT with knowledge error $\kappa(\lambda, q)$, where*

$$\kappa(\lambda, q) = (q+1) \cdot \kappa_\Pi(\lambda) + \mathbf{Adv}^{\textsf{DL-REL}}_{\mathsf{GGen}}(\mathcal{B})$$

where $\kappa_\Pi(\lambda) = Er((k_1, \ldots, k_r); \mathsf{Ch})$, and $\mathsf{Ch} = \mathbb{Z}_p^*$. Moreover, the extractor $\mathcal{E}$ terminates after making an expected number of at most $K + q \cdot (K-1)$ queries to $\mathcal{A}$, where $K = \prod_{i=1}^r k_i$ and the asymptotic running time of $\mathcal{B}$ is equivalent to that of $\mathcal{E}$.

3 Simulation-Extractability from Weak Unique Response

In this section, we introduce relaxed notions for the *unique response* property and show that along with NIZK and FS-EXT, it implies simulation-extractability (FS-SIM-EXT). Later, in Section 4, we show that BP satisfies our definition for unique response. First, we define FS-SIM-EXT and weak unique response. Then, we present our generic result for FS-SIM-EXT from NIZK, FS-EXT and weak unique response.

3.1 Simulation-Extractability in ROM

At a high-level, the simulation-extractability (FS-SIM-EXT) property ensures that extractability holds even when the adversary sees simulated proofs. The differences with FS-EXT are highlighted in orange. The definition below follows the formulation of [FKMV12]. The difference is that we allow an extractor to run in *expected* polynomial time instead of strict polynomial time. This is to base our generic theorem on the analysis of [AFK21].

Definition 3.1 (FS-SIM-EXT security). *Consider a non-interactive proof system $\Pi_{\mathsf{FS}} = (\mathsf{Setup}_{\mathsf{FS}}, \mathcal{P}_{\mathsf{FS}}, \mathcal{V}_{\mathsf{FS}})$ for relation $\mathcal{R}$ with a NIZK simulator $\mathcal{S}_{\mathsf{FS}}$. Let $(\mathcal{S}_1, \mathcal{S}'_2)$ be wrapper oracles for $\mathcal{S}_{\mathsf{FS}}$ (see Section 2.6). Π_{FS} is* simulation-extractable *(FS-SIM-EXT) with knowledge error $\hat{\kappa} : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to [0, 1]$ and with respect to $\mathcal{S}_{\mathsf{FS}}$ in the random oracle model, if there exists an extractor $\mathcal{E}$ and some polynomial* poly*, such that for any* PPT *adversary $\mathcal{P}$ that makes at most q_1 queries to $\mathcal{S}_1$ and q_2 queries to $\mathcal{S}'_2$, it holds that*

$$\widehat{\mathrm{ext}}(\mathcal{P}, \mathcal{E}, \mathcal{S}_{\mathsf{FS}}) \geq \frac{\widehat{\mathrm{acc}}(\mathcal{P}, \mathcal{S}_{\mathsf{FS}}) - \hat{\kappa}(\lambda, q_1, q_2)}{\mathsf{poly}(\lambda)}$$

and $\mathcal{E}$ halts in an expected number of steps that is polynomial in λ, q_1 and q_2, where the probabilities $\widehat{\mathrm{acc}}$ (taken over the random coins used by $\mathsf{Setup}_{\mathsf{FS}}$, $\mathcal{P}$, and $\mathcal{S}_{\mathsf{FS}}$) and $\widehat{\mathrm{ext}}$ (taken over the random coins used by $\mathsf{Setup}_{\mathsf{FS}}$, $\mathcal{P}$, $\mathcal{S}_{\mathsf{FS}}$, and $\mathcal{E}$) are defined as follows.

$$\widehat{\mathrm{acc}}(\mathcal{P}, \mathcal{S}_{\mathsf{FS}}) := \Pr\left[b = 1 \wedge (x, \mathcal{T}) \notin \mathcal{Q}_2 \;:\; (\mathsf{pp}, \mathsf{H}) \leftarrow \mathsf{Setup}_{\mathsf{FS}}(1^\lambda); (x, \mathcal{T}) \leftarrow \mathcal{P}^{\mathcal{S}_1, \mathcal{S}_2'}(\mathsf{pp}; \rho); b \leftarrow \mathcal{V}_{\mathsf{FS}}^{\mathcal{S}_1}(\mathsf{pp}, x, \mathcal{T})\right]$$

$$\widehat{\mathrm{ext}}(\mathcal{P}, \mathcal{E}, \mathcal{S}_{\mathsf{FS}}) := \Pr\left[b = 1 \;\wedge\; (x, w) \in \mathcal{R}_{\mathsf{pp}} \wedge (x, \mathcal{T}) \notin \mathcal{Q}_2 \;:\; \begin{array}{l} (\mathsf{pp}, \mathsf{H}) \leftarrow \mathsf{Setup}_{\mathsf{FS}}(1^\lambda); (x, \mathcal{T}) \leftarrow \mathcal{P}^{\mathcal{S}_1, \mathcal{S}_2'}(\mathsf{pp}; \rho); \\ b \leftarrow \mathcal{V}_{\mathsf{FS}}^{\mathcal{S}_1}(\mathsf{pp}, x, \mathcal{T}); w \leftarrow \mathcal{E}^{\mathcal{P}}(\mathsf{pp}, x, \mathcal{T}, \rho, \mathcal{Q}_1, \mathcal{Q}_2) \end{array}\right]$$

where $\mathcal{Q}_1 = \{\mathcal{Q}_{1,i}\}_{i\in[1,r]}$ is a set of query response pairs corresponding to queries to $\mathcal{S}_1$ with random oracle index i; $\mathcal{Q}_2$ is a set of statement-transcript pairs $(x, \widetilde{\mathcal{T}})$, where x is a statement queried to the proof simulation oracle $\mathcal{S}_2'$ by $\mathcal{P}$, and $\widetilde{\mathcal{T}}$ is the corresponding simulated transcript, respectively. In the experiment of $\widehat{\mathrm{ext}}$, $\mathcal{E}$ has oracle access to the next-message function of $\mathcal{P}$.

3.2 Weak Unique Response Properties

We now introduce two new definitions for the *weak* unique response property: one for interactive and the other for non-interactive protocols. Our supporting claim (Lemma 3.1) below relates the two notions, which might be of independent interest.

State-Restoration weak unique response. Our first definition considers the game $\mathsf{SR\text{-}WUR}_{\Pi}^{\mathcal{A},\mathcal{S}}(\lambda)$ in Figure 2. As the name suggests, the formalism of this notion is inspired from state-restoration witness extended emulation definitions [GT21] [BCS16]. Intuitively, state-restoration witness extended emulation says that having resettable access to the verifier (or "restoring its state", hence the name) should not help a malicious prover in producing a valid proof without knowing a witness for the statement. On a similar note, our definition SR-WUR captures the security goal that resettable access to the verifier does not help a malicious prover in generating a non-unique accepting response. Compared to the the usual UR definition for interactive protocols, SR-WUR is both stronger (in the sense that an adversary can rewind the verifier) and weaker (in the sense that an adversary is forced to use the simulated transcript to find a forgery).

Concretely, the prover initially generates an instance x on which it attempts to break the unique response property. We capture the power of the prover to rewind the verifier with an oracle $\mathbf{O}_{\mathsf{ext}}$. Roughly, the oracle allows the prover to build an execution tree, which is extended with each query to it by the prover. The prover succeeds if it comes up with another accepting transcript $\mathcal{T}$ that is part of the execution tree and have a prefix in common with the simulated transcript $\widetilde{\mathcal{T}}$. Let $\mathcal{T} = (a_1, c_1, \ldots, a_r, c_r, a_{r+1})$ denote a transcript. We write $\mathcal{T}|_i$ to denote a partial transcript consisting of the first $2i$ messages of $\mathcal{T}$, i.e., $\mathcal{T}|_i = (a_1, c_1, \ldots, a_i, c_i)$.

We remark that, our SR-WUR is presented in a non-adaptive flavor (where the statement x is fixed in the beginning) to prove subsequent lemmas with a weaker assumption. The reductions we present later will go through even though the resulting simulation-extractability claim has an adaptive flavor, by accepting a loss proportional to the number of simulation queries due to a standard guessing argument.

Definition 3.2 (SR-WUR). *Consider a $(2r+1)$-round public-coin interactive proof system $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ that has perfect HVZK simulator $\mathcal{S}$. Π is said to have* state-restoration weak unique response *(SR-WUR) with respect to a simulator $\mathcal{S}$, if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage $\mathbf{Adv}_{\Pi}^{\mathsf{SR\text{-}WUR}}(\mathcal{A}, \mathcal{S}) :=$ $\Pr[\mathsf{SR\text{-}WUR}_{\Pi}^{\mathcal{A},\mathcal{S}}(\lambda) = 1]$ is $\mathsf{negl}(\lambda)$.*

Fiat-Shamir weak unique response. We now present a similar definition tailored to non-interactive protocols. While typical unique response properties in the literature are defined for interactive protocols, [GKK+22, Definition 7] is in the non-interactive setting. Our definition below is strictly weaker than theirs, as we only need to guarantee the hardness of finding another accepting transcript forked from a simulated (honest) one.

Definition 3.3 (FS-WUR). *Consider a $(2r+1)$-round public-coin interactive proof system $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ and the resulting non-interactive system $\Pi_{\mathsf{FS}} = (\mathsf{Setup}_{\mathsf{FS}}, \mathcal{P}_{\mathsf{FS}}, \mathcal{V}_{\mathsf{FS}})$ via Fiat-Shamir transform. Let $\mathcal{S}_{\mathsf{FS}}$ be a canonical NIZK simulator for Π_{FS} (Definition 2.7) with wrapper oracles $(\mathcal{S}_1, \mathcal{S}_2')$ as defined in Section 2.6. Π_{FS} is said to have* weak unique responses *(FS-WUR) with respect to $\mathcal{S}_{\mathsf{FS}}$ if given a transcript $\widetilde{\mathcal{T}} = (\tilde{a}_1, \tilde{c}_1, \ldots, \tilde{a}_r, \tilde{c}_r, \tilde{a}_{r+1})$ simulated by $\mathcal{S}_{\mathsf{FS}}$, it is hard to find another accepting transcript $\mathcal{T} = (a_1, c_1, \ldots, a_r, c_r, a_{r+1})$ that both have a common prefix up to the ith challenge for an instance x.*

```
Game $\textsf{SR-WUR}_{\Pi}^{\mathcal{A},\mathcal{S}}(\lambda)$
    $\mathsf{win} \leftarrow \mathrm{false}$
    $\mathsf{tr} \leftarrow \epsilon$
    $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
    $(x, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}_1(\mathsf{pp})$
    $\widetilde{\mathcal{T}} \leftarrow \mathcal{S}(\mathsf{pp}, x)$
    for $i = 1, \ldots, r$ do
        $\mathsf{tr} \leftarrow \mathsf{tr} \cup \left\{ \widetilde{\mathcal{T}}|_i \right\}$
    end for
    Run $\mathcal{A}_2^{\mathbf{O}_{\mathsf{ext}}}(\widetilde{\mathcal{T}}, \mathsf{st}_{\mathcal{A}})$
    return $\mathsf{win}$
```

```
Oracle $\mathbf{O}_{\mathsf{ext}}(\mathcal{T}, a_i)$
    Parse $\mathcal{T}$ as $\mathcal{T} = (a_1, c_1, \cdots, a_{i-1}, c_{i-1})$
    if $\mathcal{T} \in \mathsf{tr}$ then
        if $i \leq r$ then
            $c_i \xleftarrow{\$} \mathsf{Ch}_i; \mathsf{tr} \leftarrow \mathsf{tr} \cup \{(\mathcal{T}, a_i, c_i)\};$
            return $c_i$
        else if $i = r + 1$ then
            $d_1 \leftarrow \mathsf{Ver}(\mathsf{pp}, x, (\mathcal{T}, a_i))$
            $d_2 \leftarrow \left(\exists j \in [1, r] : \mathcal{T}|_j = \widetilde{\mathcal{T}}|_j \wedge a_{j+1} \neq \tilde{a}_{j+1}\right)$
            if $d_1 = 1 \wedge d_2 = 1$ then
                $\mathsf{win} \leftarrow \mathrm{true}$
            return $d_1$
    return $\perp$
```

Fig. 2: State-Restoration Weak Unique Response Game

That is, for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ the advantage $\mathbf{Adv}_{\Pi_{\mathsf{FS}}}^{\textsf{FS-WUR}}(\mathcal{A}, \mathcal{S}_{\mathsf{FS}})$ defined as the following probability is $\mathsf{negl}(\lambda)$:

$$\Pr\left[\begin{array}{l} \mathcal{V}_{\mathsf{FS}}^{\mathcal{S}_1}(\mathsf{pp}, x, \mathcal{T}) = 1 \\ \wedge \left(\exists j \in [1, r] : \mathcal{T}|_j = \widetilde{\mathcal{T}}|_j \ \wedge \ a_{j+1} \neq \tilde{a}_{j+1}\right) \end{array} : \begin{array}{l} (\mathsf{pp}, \mathsf{H}) \leftarrow \mathsf{Setup}_{\mathsf{FS}}(1^\lambda); (x, \mathsf{st}) \leftarrow \mathcal{A}_1^{\mathcal{S}_1}(\mathsf{pp}); \\ \widetilde{\mathcal{T}} \leftarrow \mathcal{S}_2'(x); \mathcal{T} \leftarrow \mathcal{A}_2^{\mathcal{S}_1}(\widetilde{\mathcal{T}}, \mathsf{st}); \end{array}\right].$$

We now show that FS-WUR of Π_{FS} reduces to SR-WUR of the interactive proof system Π. Informally, the lemma below guarantees that one can construct an adversary breaking unique response in the interactive setting, given an adversary breaking weak unique response in the non-interactive setting, as long as it makes RO queries for the accepting transcript in right order.

Lemma 3.1. *Consider a $(2r+1)$-round public-coin interactive proof system $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ and the resulting non-interactive system $\Pi_{\mathsf{FS}} = (\mathsf{Setup}_{\mathsf{FS}}, \mathcal{P}_{\mathsf{FS}}, \mathcal{V}_{\mathsf{FS}})$ via Fiat-Shamir transform. Let $\mathcal{S}$ be a perfect HVZK simulator for Π and $\mathcal{S}_{\mathsf{FS}}$ be the corresponding canonical NIZK simulator for Π_{FS}. If Π has SR-WUR with respect to $\mathcal{S}$, then Π_{FS} has FS-WUR with respect to $\mathcal{S}_{\mathsf{FS}}$. That is, for every PPT adversary $\mathcal{A}$ against FS-WUR of Π_{FS} that makes q queries to $\mathcal{S}_1$, there exists a PPT adversary $\mathcal{B}$ against SR-WUR of Π such that,*

$$\mathbf{Adv}_{\Pi_{\mathsf{FS}}}^{\textsf{FS-WUR}}(\mathcal{A}, \mathcal{S}_{\mathsf{FS}}) \leq \mathbf{Adv}_{\Pi}^{\textsf{SR-WUR}}(\mathcal{B}, \mathcal{S}) + \frac{q+1}{|\mathsf{Ch}_{i_0}|}$$

where $i_0 \in [1, r]$ is the round with the smallest challenge set Ch_{i_0}. Moreover, $\mathcal{B}$ makes at most q queries to its oracle and is nearly as efficient as $\mathcal{A}$.

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be the NIZK prover against FS-WUR. We define an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against SR-WUR. Without loss of generality we assume that $\mathcal{A}$ does not repeat the same RO queries.

In the first stage, $\mathcal{B}_1$ will run $\mathcal{A}_1$ on pp. Whenever $\mathcal{A}_1$ makes RO queries, $\mathcal{B}_1$ invokes $\mathcal{S}_1$ and keeps track of the query history in tables $\mathcal{Q}_1 = \{\mathcal{Q}_{1,1}, \ldots, \mathcal{Q}_{1,r}\}$. When $\mathcal{A}_1$ returns $(x, \mathsf{st}_{\mathcal{A}})$, $\mathcal{B}_1$ outputs $(x, \mathsf{st}_{\mathcal{B}})$ where $\mathsf{st}_{\mathcal{B}} = (\mathsf{pp}, \mathsf{st}_{\mathcal{A}}, x, \mathcal{Q})$.

In the second stage, $\mathcal{B}_2$ is given $\mathsf{st}_{\mathcal{B}}$ and $\widetilde{\mathcal{T}}$, which is the output on invoking the HVZK simulator $\mathcal{S}$ on x. Then, $\mathcal{B}_2$ programs the RO tables such that $\mathcal{Q}_{1,1}[\mathsf{pp}, x, \tilde{a}_1] := \tilde{c}_1, \ldots, \mathcal{Q}_{1,r}[\mathsf{pp}, x, \tilde{a}_1, \tilde{c}_1, \ldots, \tilde{a}_r] := \tilde{c}_r$. Now, $\mathcal{B}_2$ invokes $\mathcal{A}_2$ on $\widetilde{\mathcal{T}}$ and $\mathsf{st}_{\mathcal{A}}$ and responds to the RO queries as follows.

- $\mathcal{B}_2$ initializes a set of partial transcripts tr as

$$\mathsf{tr} = \{(\tilde{a}_1, \tilde{c}_1), \ldots, (\tilde{a}_1, \tilde{c}_1, \ldots, \tilde{a}_r, \tilde{c}_r)\}.$$

- Whenever $\mathcal{B}_2$ receives from $\mathcal{A}_2$ a query to $\mathcal{S}_1$ of the form $((\mathsf{pp}, x, \mathcal{T}, a_i), i)$ where transcript $\mathcal{T} = (a_1, c_1, \ldots, a_{i-1}, c_{i-1})$, it checks if $\mathcal{T} \in \mathsf{tr}$. If that is the case, it queries $\mathbf{O}_{\mathsf{ext}}$ with $\mathcal{T}$ and a_i (with representation). On receiving challenge c_i, it updates tr such that $\mathsf{tr} \leftarrow \mathsf{tr}||(\mathcal{T}, a_i, c_i)$ and programs the RO table such that $\mathcal{Q}_{1,i}[\mathsf{pp}, x, \mathcal{T}, a_i] := c_i$. Note that the partial transcript set tr constructed as above is guaranteed to be identical to that of SR-WUR.

– When $\mathcal{B}_2$ receives from $\mathcal{A}_2$ a forged transcript $\mathcal{T}$ at the end, it checks whether the winning condition is satisfied, i.e.,

$$\mathcal{V}_{\mathsf{FS}}^{\mathcal{S}_1}(\mathsf{pp}, x, \mathcal{T}) = 1 \wedge (\exists j \in [1, r] : \mathcal{T}|_j = \widetilde{\mathcal{T}}|_j \wedge a_{j+1} \neq \tilde{a}_{j+1}).$$

If that is the case but $\mathcal{T}|_i \notin \mathsf{tr}$ for some $i \in [1, r]$, $\mathcal{B}_2$ aborts, since it implies $\mathcal{T}$ is not present in the partial transcript set maintained by $\mathbf{O}_{\mathsf{ext}}$ and therefore does not qualify as a winning transcript in the SR-WUR game. Otherwise, it forwards $\mathcal{T}$ to $\mathbf{O}_{\mathsf{ext}}$.

If $\mathcal{B}_2$ does not abort, $\mathcal{B}_2$ wins the SR-WUR game because all the partial transcripts of $\mathcal{T}$ are recorded by $\mathbf{O}_{\mathsf{ext}}$ in the SR-WUR game. We now bound the probability that $\mathcal{B}_2$ aborts. It aborts only if (1) the accepting transcript $\mathcal{T}$ was crafted by $\mathcal{A}_2$ without querying the RO in order, or (2) the accepting transcript $\mathcal{T}$ contains some challenge that was obtained without querying the RO at all.

The first case happens only if for some $i, j \in [1, r]$ such that $j < i$, a tuple $(\mathsf{pp}, x, \mathcal{T}|_{i-1}, a_i)$ was queried to $\mathcal{S}_1$ *before* $(\mathsf{pp}, x, \mathcal{T}|_{j-1}, a_j)$ was queried. In that case, when $(\mathsf{pp}, x, \mathcal{T}|_{j-1}, a_j)$ is queried later, $\mathcal{S}_1$ must return c_j already present in $\mathcal{T}|_{i-1}$ for $\mathcal{T}$ to be accepting. This happens with probability at most $q/|\mathsf{Ch}_j| \leq q/|\mathsf{Ch}_{i_0}|$ (where i_0 is the round with the smallest Ch_{i_0}) since the adversary can try to guess any challenge value for at most q times.

The second case happens only if for some $i \in [1, r]$, a tuple $(\mathsf{pp}, x, \mathcal{T}|_{i-1}, a_i)$ was never queried to $\mathcal{S}_1$ by $\mathcal{A}$. In that case, when $(\mathsf{pp}, x, \mathcal{T}|_{i-1}, a_i)$ is queried later, $\mathcal{S}_1$ must return c_i already present in $\mathcal{T}$ for $\mathcal{T}$ to be accepting. This happens with probability at most $1/|\mathsf{Ch}_i| \leq 1/|\mathsf{Ch}_{i_0}|$ since $\mathcal{V}_{\mathsf{FS}}$ queries $\mathcal{S}_1$ once for each challenge.

Overall, $\mathcal{B}$ wins whenever $\mathcal{A}$ wins except with probability at most $(q + 1)/|\mathsf{Ch}_{i_0}|$.

□

3.3 From Weak Unique Response to Simulation-Extractability

In this part, we prove that the weak unique response property is indeed sufficient for upgrading plain extractability to simulation-extractability. Intuitively, one can construct a prover $\mathcal{P}$ that outputs a proof in the FS-EXT game by internally invoking $\hat{\mathcal{P}}$ playing the FS-SIM-EXT game. $\mathcal{P}$ answers $\mathcal{S}_2'$ queries by running $\mathcal{S}_{\mathsf{FS}}$ and forwards $\mathcal{S}_1$ queries made by $\hat{\mathcal{P}}$ to the actual random oracle H *except for the inputs already programmed by* $\mathcal{S}_{\mathsf{FS}}$. Notice that the programmed RO entries introduce inconsistencies between the responses of $\mathcal{S}_1$ and H. Hence, $\mathcal{P}$ aborts if $\hat{\mathcal{P}}$ outputs a transcript that shares some prefix with any of the simulated transcripts. The probability that $\hat{\mathcal{P}}$ aborts can be bounded by a negligible function, assuming the FS-WUR property. Unless $\mathcal{P}$ aborts, we can run a FS-EXT extractor against $\mathcal{P}$ to obtain a valid witness in expected polynomial time.

Lemma 3.2. *Consider a non-interactive proof system Π_{FS} with a canonical NIZK simulator $\mathcal{S}_{\mathsf{FS}}$. If Π_{FS} is FS-EXT with negligible knowledge error and is FS-WUR, then it is FS-SIM-EXT with negligible knowledge error. Concretely, if Π_{FS} is FS-EXT with knowledge error κ, then there exists a* PPT *adversary $\mathcal{A}$ against FS-WUR such that Π_{FS} is FS-SIM-EXT with knowledge error $\hat{\kappa}$ with respect to $\mathcal{S}_{\mathsf{FS}}$, where*

$$\hat{\kappa}(\lambda, q_1, q_2) = q_2 \cdot \mathbf{Adv}_{\Pi_{\mathsf{FS}}}^{\textit{FS-WUR}}(\mathcal{A}, \mathcal{S}_{\mathsf{FS}}) + \kappa(\lambda, q_1)$$

and q_1 (resp. q_2) is the number of queries to $\mathcal{S}_1$ (resp. $\mathcal{S}_2'$).

Proof. Let $\mathcal{E}$ be a FS-EXT extractor. We construct a FS-SIM-EXT extractor $\hat{\mathcal{E}}$. Given an arbitrary FS-SIM-EXT prover $\hat{\mathcal{P}}$, $\hat{\mathcal{E}}$ converts $\hat{\mathcal{P}}$ into a FS-EXT prover $\mathcal{P}$ that only has access to H.

To this end, let us first describe simple hybrids.

– $\mathsf{G}_0(\hat{\mathcal{P}}, \mathcal{S}_{\mathsf{FS}})$: This game first runs a prover $\hat{\mathcal{P}}$ on input $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ with access to $\mathcal{S}_1, \mathcal{S}_2'$. On receiving output $(x, \mathcal{T})$ from $\hat{\mathcal{P}}$, G_0 outputs 1 if and only if $1 = \mathcal{V}_{\mathsf{FS}}^{\mathcal{S}_1}(\mathsf{pp}, x, \mathcal{T})$ and $(x, \mathcal{T}) \notin \mathcal{Q}_2$. By definition we have that

$$\Pr[\mathsf{G}_0(\hat{\mathcal{P}}, \mathcal{S}_{\mathsf{FS}}) = 1] = \widehat{\mathrm{acc}}(\hat{\mathcal{P}}, \mathcal{S}_{\mathsf{FS}}).$$

– $\mathsf{G}_1(\hat{\mathcal{P}}, \mathcal{S}_{\mathsf{FS}})$: This game is identical to the previous one, except that it aborts after setting the flag *bad* if $(x, \mathcal{T})$ output by the adversary shares its prefix with some simulated statement-transcript pair $(x, \widetilde{\mathcal{T}}) \in \mathcal{Q}_2$, i.e., there exists some $j \in [1, r]$ such that $\mathcal{T}|_j = \widetilde{\mathcal{T}}|_j$ *and* $a_{j+1} \neq \tilde{a}_{j+1}$. Since G_0 and G_1 are identical until *bad* is set, we have that

$$|\Pr[\mathsf{G}_0(\hat{\mathcal{P}}, \mathcal{S}_{\mathsf{FS}}) = 1] - \Pr[\mathsf{G}_1(\hat{\mathcal{P}}, \mathcal{S}_{\mathsf{FS}}) = 1]| \leq \Pr[\mathsf{G}_1(\hat{\mathcal{P}}, \mathcal{S}_{\mathsf{FS}}) \text{ sets } bad]. \quad (3)$$

Later we bound (3) by $q_2 \cdot \mathbf{Adv}_{\Pi_{\mathsf{FS}}}^{\mathsf{FS\text{-}WUR}}(\mathcal{A}, \mathcal{S}_{\mathsf{FS}})$.

Algorithm 4: Construction of $\mathcal{P}$

```
P^H(pp)
 1: Initialize $\mathsf{st}$ with empty $\mathcal{Q}_1 = \{\mathcal{Q}_{1,1}, \ldots, \mathcal{Q}_{1,r}\}$
 2: $\mathcal{Q}_2 := \varnothing$
 3: $(x, \mathcal{T}) \leftarrow \hat{\mathcal{P}}^{\mathcal{S}_1, \mathcal{S}_2'}(\mathsf{pp})$
 4: $b \leftarrow \mathcal{V}_{\mathsf{FS}}^{\mathcal{S}_1}(\mathsf{pp}, x, \mathcal{T})$
 5: if $(b = 0) \vee ((x, \mathcal{T}) \in \mathcal{Q}_2)$ then
 6:     return $(\bot, \bot)$
 7: end if
 8: if $\exists (x, \widetilde{\mathcal{T}}) \in \mathcal{Q}_2 : (\exists j \in [1, r] : \mathcal{T}|_j = \widetilde{\mathcal{T}}|_j \wedge a_{j+1} \neq \tilde{a}_{j+1})$ then
 9:     $bad :=$ true
10:     return $(\bot, \bot)$
11: end if
12: return $(x, \mathcal{T})$

S_1(t, i)
 1: Retrieve $\mathcal{Q}_{1,i}$ from $\mathsf{st}$
 2: if $\mathcal{Q}_{1,i}[t] \neq \bot$ then
 3:     return $\mathcal{Q}_{1,i}[t]$
 4: else
 5:     $c_i := \mathsf{H}_i(t)$
 6:     $\mathcal{Q}_{1,i}[t] := c_i$
 7:     Update $\mathcal{Q}_{1,i}$ in $\mathsf{st}$
 8:     return $\mathcal{Q}_{1,i}[t]$
 9: end if

S_2'(x)
 1: $(\widetilde{\mathcal{T}}, \mathsf{st}) \leftarrow \mathcal{S}_{\mathsf{FS}}(2, \mathsf{st}, x)$ // $\mathcal{Q}_1$ in st gets updated by $\mathcal{S}_{\mathsf{FS}}$
 2: $\mathcal{Q}_2 := \mathcal{Q}_2 \cup \{(x, \widetilde{\mathcal{T}})\}$
 3: return $\widetilde{\mathcal{T}}$
```

Construction of $\mathcal{P}$ for FS-EXT. We now construct a FS-EXT adversary $\mathcal{P}$, detailed in Alg. 4. The differences with the procedures of $\widehat{\text{acc}}$ are highlighted in orange. On a high level, $\mathcal{P}$ simulates the view of $\hat{\mathcal{P}}$ in G_1 while obtaining responses of $\mathcal{S}_1$ by querying the random oracle H. In doing so, the simulated random oracle responses in $\mathcal{Q}_{1,i}$ for $i \in [1, r]$ are identical to those of H, *except for the ones programmed inside* $\mathcal{S}_2'$. Since the prefix check performed on Line 8 rules out a forgery pair $(x, \mathcal{T})$ involving simulated random oracle entries, $\mathcal{P}$ is guaranteed to output $(x, \mathcal{T})$ that is also accepting with respect to H, i.e., $\mathsf{H}_i(\mathsf{pp}, x, a_1, c_1, \ldots, a_i) = c_i$ for $i \in [1, r]$. Hence, we have that

$$\Pr[\mathsf{G}_1(\hat{\mathcal{P}}, \mathcal{S}_{\mathsf{FS}}) = 1] = \text{acc}(\mathcal{P}).$$

Construction of $\hat{\mathcal{E}}$ for FS-SIM-EXT. We are now ready to construct a FS-SIM-EXT extractor $\hat{\mathcal{E}}$. By assumption, there exists an expected polynomial time FS-EXT extractor $\mathcal{E}$ satisfying Definition 2.8. After the first run of $\hat{\mathcal{P}}$, $\hat{\mathcal{E}}$ obtains $(\mathsf{pp}, x, \mathcal{T}, \mathcal{Q}_1, \mathcal{Q}_2)$ as input. If there exists $(x, \widetilde{\mathcal{T}}) \in \mathcal{Q}_2$ such that $\mathcal{T} \neq \widetilde{\mathcal{T}}$ and $\mathcal{T}|_j = \widetilde{\mathcal{T}}|_j$ for some $j \in [1, r]$, then $\hat{\mathcal{E}}$ aborts. Otherwise, $\hat{\mathcal{E}}$ strips from $\mathcal{Q}_1$ the entries involving programmed partial transcripts derived from $\mathcal{Q}_2$, and passes $(\mathsf{pp}, x, \mathcal{T}, \mathcal{Q}_1)$ to $\mathcal{E}$. Whenever $\mathcal{E}$ requests running a prover, $\hat{\mathcal{E}}$ gives $\mathcal{E}$ oracle access to the next-message function of $\mathcal{P}$ in Alg 4, which internally calls the next-message function of $\hat{\mathcal{P}}$. In this way, whenever $\mathcal{E}$ outputs a valid witness, $\hat{\mathcal{E}}$ also succeeds in doing so. Putting everything together, we have

$$\begin{aligned} \widehat{\text{ext}}(\hat{\mathcal{P}}, \hat{\mathcal{E}}, \mathcal{S}_{\mathsf{FS}}) \geq \text{ext}(\mathcal{P}, \mathcal{E}) &\geq \frac{\text{acc}(\mathcal{P}) - \kappa(\lambda, q_1)}{\mathsf{poly}(\lambda)} \\ &\geq \frac{\widehat{\text{acc}}(\hat{\mathcal{P}}, \mathcal{S}_{\mathsf{FS}}) - \Pr[\mathsf{G}_1(\hat{\mathcal{P}}, \mathcal{S}_{\mathsf{FS}}) \text{ sets } bad] - \kappa(\lambda, q_1)}{\mathsf{poly}(\lambda)} \end{aligned}$$

By inspection, we observe the expected running time $\text{Time}(\hat{\mathcal{E}}) \approx q_2 \cdot \text{Time}(\mathcal{S}_{\mathsf{FS}}) \cdot \text{Time}(\mathcal{E})$, which is clearly polynomial in λ, q_1 and q_2 if $\text{Time}(\mathcal{E})$ is polynomial in λ and q_1.

Reduction to FS-WUR. We now bound the probability (3) that the game G_1 sets bad. We argue that, if there exists $\hat{\mathcal{P}}$ that causes $\mathsf{G}_1(\hat{\mathcal{P}}, \mathcal{S}_{\mathsf{FS}})$ to abort (or in other words, that causes $\mathcal{P}$ to abort), one can use $\hat{\mathcal{P}}$ to construct another adversary $\mathcal{A}$ that breaks FS-WUR with respect to $\mathcal{S}_{\mathsf{FS}}$. The reduction goes as follows. The differences with G_1 are highlighted in orange.

– $\mathcal{A}$ first picks a query index $k \in [1, q_2]$ uniformly at random.

– On receiving pp in the FS-WUR game, $\mathcal{A}$ forwards pp to $\hat{\mathcal{P}}$.

– Whenever $\hat{\mathcal{P}}$ makes a simulation query with input x, if this is the kth simulation query then it forwards x to $\mathcal{S}_2'$ in the FS-WUR. We denote the statement-transcript pair of the kth query by $(x^k, \widetilde{\mathcal{T}}^k)$. Otherwise, $\mathcal{A}$ internally invokes $\mathcal{S}_{\mathsf{FS}}(2, \mathsf{st}, x)$ to obtain $(\widetilde{\mathcal{T}}, \mathsf{st}')$. It records a statement-proof pair $(x, \widetilde{\mathcal{T}})$ in the set $\mathcal{Q}_2$. Then it programs the RO tables $\mathcal{Q}_1$ for every challenge in $\widetilde{\mathcal{T}}$ as $\mathcal{S}_{\mathsf{FS}}(2, \mathsf{st}, x)$ would do.

– Whenever $\hat{\mathcal{P}}$ (or $\mathcal{V}_{\mathsf{FS}}$ at the end) makes a random oracle query with input $((\mathsf{pp}, x, \mathcal{T}, a_i), i)$, where $\mathcal{T} = (a_1, c_1, \ldots, a_{i-1}, c_{i-1})$, $\mathcal{A}$ checks whether $(x^k, \widetilde{\mathcal{T}}^k)$ has prefix in common with $\mathcal{T}$, i.e., for some $1 \le j \le i-1$ it holds that $\mathcal{T}|_j = \widetilde{\mathcal{T}}^k|_j$. If that is the case, it forwards the query $((\mathsf{pp}, x, \mathcal{T}, a_i), i)$ to $\mathcal{S}_1$ in the FS-WUR game, receives $c_i \in \mathsf{Ch}_i$, and updates $\mathcal{Q}_{1,i}$ accordingly. Otherwise, it lazily samples c_i from Ch_i and updates $\mathcal{Q}_{1,i}$ accordingly, as $\mathcal{S}_{\mathsf{FS}}(1, st, (\mathsf{pp}, x, \mathcal{T}, a_i), i)$ would do.
– When $\hat{\mathcal{P}}$ outputs a forgery $(x, \mathcal{T})$, $\mathcal{A}$ first checks whether it would set *bad* in the game G_1, i.e., for some simulated statement-transcript pair $(x, \widetilde{\mathcal{T}}) \in \mathcal{Q}_2$, there exists some $j \in [1, r]$ such that $\mathcal{T}|_j = \widetilde{\mathcal{T}}|_j$ *and* $a_{j+1} \neq \tilde{a}_{j+1}$. If that is the case, $\mathcal{A}$ forwards $\mathcal{T}$ to the FS-WUR game as a forgery.

The above procedure perfectly simulates $\hat{\mathcal{P}}$'s view in the game G_1. By construction $\mathcal{A}$ breaks FS-WUR with respect to $\mathcal{S}_{\mathsf{FS}}$ if (G_1 sets *bad*) *and* ($x = x^k \wedge \mathcal{T}$ has some shared prefix with $\widetilde{\mathcal{T}}^k$), because then it is guaranteed that for every $i \in [1, r]$, c_i has been obtained by querying the oracles $(\mathcal{S}_1, \mathcal{S}_2')$ in the FS-WUR game. Therefore, $\mathcal{T}$ does qualify as a valid forgery in the FS-WUR game. Conditioned on the event that G_1 sets *bad*, the probability that $\mathcal{A}$ wins is at least $1/q_2$. Therefore, we have

$$\frac{1}{q_2} \cdot \Pr[\mathsf{G}_1(\hat{\mathcal{P}}, \mathcal{S}_{\mathsf{FS}}) \text{ sets } bad] \le \mathbf{Adv}^{\mathsf{FS\text{-}WUR}}_{\Pi_{\mathsf{FS}}}(\mathcal{A}, \mathcal{S}_{\mathsf{FS}}).$$

□

4 Non-Malleability of Bulletproofs – Arithmetic Circuits

The Bulletproof (henceforth referred as BP) protocol for arithmetic circuit satisfiability (see relation 2) is formally described in Protocol 1[12] and proceeds as follows. The algorithm $\mathsf{Setup}(1^\lambda)$ parameterized by n, Q invokes a group parameter generator $\mathsf{GGen}(1^\lambda)$ to obtain $\mathbb{G}$, samples $(g, h, u, \mathbf{g}, \mathbf{h}) \xleftarrow{\$} \mathbb{G}^{2n+3}$, defines the challenge space $\mathsf{Ch} = \mathbb{Z}_p^*$, and outputs $\mathsf{pp} := (n, Q, g, h, u, \mathbf{g}, \mathbf{h})$. In the first round, the prover commits to values on the wire of the circuit (i.e. $\mathbf{a}_L, \mathbf{a}_R$ and $\mathbf{a}_O$), and the blinding vectors $(\mathbf{s}_L, \mathbf{s}_R)$. It receives challenges y, z from the verifier. Based on these challenges, the prover defines three polynomials, l, r and t, where $t(X) = \langle l(X), r(X) \rangle$, and commits to the coefficients of the polynomial t in the third round, i.e. commitments T_1, T_3, T_4, T_5, and, T_6[13]. On receiving a challenge x from the verifier, the prover evaluates polynomials l, r on this challenge point, computes $\hat{t} = \langle l(x), r(x) \rangle$, and values β_x, μ, and sends $\beta_x, \mu, \hat{t}, \mathbf{l} = l(x)$ and $\mathbf{r} = r(x)$ in the fifth round. The verifier accepts if: the commitments $\{T_i\}_{i \in \mathscr{S}}$ (for $\mathscr{S} = \{1, 3, 4, 5, 6\}$) are to the correct polynomial t and if $\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle$. To get logarithmic proof size, the prover and verifier define an instance of the inner dot product for checking the condition $\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle$, instead of sending vectors $\mathbf{l}, \mathbf{r}$ in clear. The inner product subroutine InPrd is presented in Appendix A. Since InPrd is a $(2m+1)$-round protocol with $m = \log(n)$, BP is a $(2r+1)$-round public-coin protocol with $r = 3 + m$.

4.1 Alternative Simulation Strategy

Let $\mathcal{S}_{\mathsf{BP}}$ denote the perfect HVZK simulator from [BBB+18]. We define an alternative simulation strategy $\mathcal{S}'_{\mathsf{BP}}$ for BP which will be used in the proof of FS-WUR. The canonical NIZK simulators corresponding to $\mathcal{S}_{\mathsf{BP}}$ and $\mathcal{S}'_{\mathsf{BP}}$ are denoted by $\mathcal{S}_{\mathsf{FS\text{-}BP}}$ and $\mathcal{S}'_{\mathsf{FS\text{-}BP}}$, respectively. The simulator $\mathcal{S}'_{\mathsf{BP}}$ essentially works as $\mathcal{S}_{\mathsf{BP}}$, except that it explicitly computes group representation for each simulated element. To do so, it will generate A_I, A_O as well as the T_i's by learning their discrete logarithm in bases g, h, u instead of generically sampling random group elements like in the original proof. This makes no difference for the ZK claim and makes the proof of FS-WUR simpler.

Remark 4.1. The simulator for the recursive version of Bulletproof e.g., the one that calls InPrd instead of sending $\mathbf{l}, \mathbf{r}$ directly, can easily be constructed from the simulator above by running the InPrd protocol on $\mathbf{l}, \mathbf{r}$. The simulator also outputs the representation for the elements L_i, R_i generated during this protocol and this representation will be used explicitly in the proof later.

[12] The protocol described in [BBB+18] is for relation that additionally incorporates a vector of Pedersen commitments $\mathbf{V}$ and the weight parameter $\mathbf{W_V}$ as part of the statement. We omit these for the sake of simplicity since they do not impact the unique response analysis.

[13] The degree two term is independent of the witness and can be computed by the verifier, therefore there is no T_2 commitment.

Protocol 1: BP

This is a protocol for the following relation

$$\mathcal{R}_{\mathsf{ACSPf}} = \left\{ \begin{array}{l} ((n, Q, g, h, u, \mathbf{g}, \mathbf{h}), (\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c}), (\mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O)) \mid \\ \mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O \wedge \mathbf{W}_L \cdot \mathbf{a}_L + \mathbf{W}_R \cdot \mathbf{a}_R + \mathbf{W}_O \cdot \mathbf{a}_R = \mathbf{c} \end{array} \right\}.$$

Prover $\mathcal{P}$ and verifier $\mathcal{V}$ interact as follows:

1. $\mathcal{P}$ samples $\mathbf{s}_L \xleftarrow{\$} \mathbb{Z}_p^n, \mathbf{s}_R \xleftarrow{\$} \mathbb{Z}_p^n, \alpha, \beta, \rho \xleftarrow{\$} \mathbb{Z}_p$. $\mathcal{P}$ sends
$$S = h^{\rho}\mathbf{g}^{\mathbf{s}_L}\mathbf{h}^{\mathbf{s}_R},\ A_I = h^{\alpha}\mathbf{g}^{\mathbf{a}_L}\mathbf{h}^{\mathbf{a}_R},\ A_O = h^{\beta}\mathbf{g}^{\mathbf{a}_O}.$$
2. $\mathcal{V}$ sends challenge $y, z \xleftarrow{\$} \mathbb{Z}_p^*$.
3. $\mathcal{P}$ samples $\beta_i \xleftarrow{\$} \mathbb{Z}_p, \forall i \in \mathscr{S}$, computes polynomials
$$\begin{aligned} l(X) &= \mathbf{a}_L X + \mathbf{a}_O X^2 + \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_R) \cdot X + \mathbf{s}_L X^3, \\ r(X) &= \mathbf{y}^n \circ \mathbf{a}_R \cdot X - \mathbf{y}^n + \mathbf{z}_{[1:]}^{Q+1} \cdot (\mathbf{W}_L \cdot X + \mathbf{W}_O) + \mathbf{y}^n \circ \mathbf{s}_R \cdot X^3, \\ t(X) &= \langle l(X), r(X) \rangle = \sum_{i=1}^{6} t_i X^i. \end{aligned}$$
$\mathcal{P}$ computes $\forall i \in \mathscr{S} : T_i = g^{t_i} h^{\beta_i}$ and sends $\{T_i\}_{i \in \mathscr{S}}$.
4. $\mathcal{V}$ sends challenge $x \xleftarrow{\$} \mathbb{Z}_p^*$.
5. $\mathcal{P}$ computes
$$\begin{aligned} &\mathbf{l} = l(x),\ \mathbf{r} = r(x),\ \hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle, \\ &\beta_x = \beta_1 \cdot x + \sum_{i=3}^{6} \beta_i \cdot x^i, \mu = \alpha \cdot x + \beta \cdot x^2 + \rho \cdot x^3 \end{aligned}$$
and sends $\hat{t}, \beta_x, \mu$.
6. $\mathcal{V}$ sends challenge $w \xleftarrow{\$} \mathbb{Z}_q^*$.
7. $\mathcal{P}, \mathcal{V}$ both compute
$$\begin{aligned} &\mathbf{h}' = \mathbf{h}^{\mathbf{y}^{-n}},\ u' = u^w, \\ &W_L = \mathbf{h}'^{\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_L},\ W_R = \mathbf{g}^{\mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_R)},\ W_O = \mathbf{h}'^{\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_O}, \\ &P = A_I^x \cdot A_O^{x^2} \cdot \mathbf{h}'^{-\mathbf{y}^n} \cdot W_L^x \cdot W_R^x \cdot W_O \cdot S^{x^3}, \\ &P' = h^{-\mu} P (u')^{\hat{t}} \end{aligned}$$
and jointly execute the inner product argument protocol as
$$\langle \mathsf{InPrd}.\mathcal{P}((\mathbf{g}, \mathbf{h}', u', P'), (\mathbf{l}, \mathbf{r})), \mathsf{InPrd}.\mathcal{V}(\mathbf{g}, \mathbf{h}', u', P') \rangle.$$
8. $\mathcal{V}$ computes
$$\begin{aligned} &\delta(y, z) = \left\langle \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_R), \mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_L \right\rangle, \\ &R = g^{x^2(\delta(y,z) + \langle \mathbf{z}_{[1:]}^{Q+1}, \mathbf{c} \rangle)} \cdot T_1^x \cdot \prod_{i=3}^{6} T_i^{x^i}, \\ &\mathsf{InPrd}.\mathcal{V}(\mathbf{g}, \mathbf{h}', u', P') \to b \end{aligned}$$
and returns 1 iff $b = 1$ and $g^{\hat{t}} h^{\beta_x} = R$.

Simulator 1: $\mathcal{S}'_{\mathsf{BP}}$

The simulator $\mathcal{S}'_{\mathsf{BP}}$ is given as input:

$$\mathsf{pp} = (n, Q, g, h, u, \mathbf{g}, \mathbf{h}),\ \mathsf{x} = (\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c})$$

The transcript is simulated as follows where the difference with the original simulator $\mathcal{S}_{\mathsf{BP}}$ is marked in orange:

```
1: $x, y, w, z \xleftarrow{\$} \mathbb{Z}_p^*$
2: $\beta_x, \mu \xleftarrow{\$} \mathbb{Z}_p$
3: $\mathbf{l}, \mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^n$
4: $\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle$
5: $\rho_I, \rho_O, t_3, t_4, t_5, t_6, \beta_3, \beta_4, \beta_5, \beta_6 \xleftarrow{\$} \mathbb{Z}_p$
6: $A_I = g^{\rho_I}, A_O = u^{\rho_O}$
7: $T_i = g^{t_i} h^{\beta_i}$ for $i \in \{3, 4, 5, 6\}$

1: $\mathbf{h}' = \mathbf{h}^{\mathbf{y}^{-n}},\ u' = u^w$
2: $W_L = \mathbf{h}'^{\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_L},\ W_R = \mathbf{g}^{\mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_R)},\ W_O = \mathbf{h}^{\mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_O)}$
3: $S = \left( A_I^x \cdot A_O^{x^2} \cdot \mathbf{g}^{-\mathbf{l}} \cdot (\mathbf{h}')^{-\mathbf{y}^n - \mathbf{r}} \cdot W_L^x \cdot W_R^x \cdot W_O \cdot h^{-\mu} \right)^{-x^{-3}}$
4: $T_1 = \left( h^{-\beta_x} \cdot g^{x^2 \cdot (\delta(y,z) + \langle \mathbf{z}_{[1:]}^{Q+1}, \mathbf{c} \rangle) - \hat{t}} \cdot \prod_{i=3}^{6} T_i^{x^i} \right)^{-x^{-1}}$
5: $\mathcal{T} = (S, A_I, A_O; y, z; \{T_i\}_{i \in \mathscr{S}}; x; \hat{t}, \beta_x, \mu; w; \mathbf{l}, \mathbf{r})$
6: Output $\mathcal{T}$
```

Claim 1. *The protocol* BP *is perfect* HVZK *with simulator* $\mathcal{S}'_{\mathsf{BP}}$.

Proof. The claim follows directly from the proof of HVZK in [BBB+18] by observing that the way $A_I, A_O, T_3, T_4, T_5, T_6$ are generated in our and their simulator produces the exact same distribution (in their case they are sampled as random elements from the group; in ours, we generate them by raising generators to random exponents, and those are not re-used anywhere else). □

4.2 Weak Unique Responses of FS-BP

The following claim is crucial for invoking our generic result from Lemma 3.2. We remind the reader that proving uniqueness of the randomized commitments T_i's is made possible thanks to our relaxed definition: if the adversary was allowed to control both transcripts, it would be trivial to break the (strong) unique response by honestly executing the prover algorithm twice with known witness and by committing to t_i using distinct randomnesses β_i and β'_i. Our proof below on the other hand argues that a cheating prover against FS-WUR has a hard time forging T_i once one of the transcripts has been fixed by a simulator. In other words, they cannot reuse parts of simulated proofs without knowing how the simulated messages were generated. This is true even for true statements where the prover might know the witness.

Proof outline. First, consider the proof idea for proving SR-WUR for BP in the proceedings version of this work [GOP+22]. In the SR-WUR game, the adversary queries a simulator $\mathcal{S}$ and receives a simulated transcript $\widetilde{\mathcal{T}}$. It then attempts to submit a transcript $\mathcal{T}$ that forks with respect to $\widetilde{\mathcal{T}}$. The central idea of the proof there is: given group representations for both the transcripts, one can analyse them to break DL-REL. In [GOP+22], this is obtained by assuming an algebraic adversary (assuming AGM) and an *algebraic simulator*. The latter means that the simulator outputs group representation along with the simulated transcript. In this work, we will follow the same high-level idea of breaking DL-REL using group representation of the transcripts, however, we will derive these representations for both the transcripts in an alternate way so that we remove the reliance on both an algebraic adversary and an algebraic simulator. Another point of difference with [GOP+22] is that we will prove FS-WUR directly.
Our first observation is that, for FS-BP, the group representation for any valid transcript $\mathcal{T}$ can be derived from the tree of transcripts corresponding to $\mathcal{T}$ (see Definition 2.9). To build the tree of transcript, we can run the knowledge extractor from [AFK21] (see Section 2.8). However, there are two caveats to this approach. First, rewinding cannot be done with respect to the simulated transcript $\widetilde{\mathcal{T}}$ as we cannot rewind the simulator. Thus, we cannot derive the group representation for $\widetilde{\mathcal{T}}$. We resolve this issue by considering a hybrid where instead of running some valid NIZK simulator $\mathcal{S}_{\mathsf{FS\text{-}BP}}$ for FS-BP, we run the specific NIZK simulator $\mathcal{S}'_{\mathsf{FS\text{-}BP}}$. This change is perfectly indistinguishable from the FS-WUR experiment played with respect to $\mathcal{S}_{\mathsf{FS\text{-}BP}}$ as both $\mathcal{S}_{\mathsf{FS\text{-}BP}}$ and $\mathcal{S}'_{\mathsf{FS\text{-}BP}}$ are canonical NIZK simulators for FS-BP derived from perfect HVZK simulators. Now, observe that $\mathcal{S}'_{\mathsf{FS\text{-}BP}}$ can output the group representation of the simulated transcript in terms of the statement and pp, in addition to the transcript itself. Note that this switch happens only in the reduction, and thus, the theorem statement holds for any valid simulator $\mathcal{S}_{\mathsf{FS\text{-}BP}}$ (as opposed to just algebraic simulators in [GOP+22]).
Second, the adversary cannot be rewinded to a point before the round where $\widetilde{\mathcal{T}}$ diverges from $\mathcal{T}$ for the first time. Doing so will result in reprogramming the random oracle entry at this point, which then will not match the challenge in $\widetilde{\mathcal{T}}$. This makes it easy for the adversary to distinguish real and ideal worlds.

Inability to rewind the adversary fully implies that the group representation for a part of the transcript $\mathcal{T}$ (the part in common with $\widetilde{\mathcal{T}}$) cannot be derived. Once again, switching to the simulator $\mathcal{S}'_{\mathsf{FS\text{-}BP}}$ in the reduction resolves this problem.

Claim 2. *Protocol* FS-BP *satisfies weak unique response (FS-WUR) with respect to a canonical NIZK simulator $\mathcal{S}_{\mathsf{FS\text{-}BP}}$ in programmable ROM, under the assumption that solving the discrete-log relation is hard. That is, for every* PPT *adversary $\mathcal{A}_{ur}$ against FS-WUR of* FS-BP *that makes at most q random oracle queries, there exists an adversary $\mathcal{B}$ against DL-REL that runs in expected polynomial time in q and λ, such that,*

$$\mathbf{Adv}^{\textit{FS-WUR}}_{\mathsf{FS\text{-}BP}}(\mathcal{A}_{ur}, \mathcal{S}_{\mathsf{FS\text{-}BP}}) \leq (1-\kappa_2)\cdot \mathbf{Adv}^{\textit{DL-REL}}_{\mathsf{GGen}}(\mathcal{B}) + \frac{2}{p-1} + (q+1)\cdot\kappa_2$$

where $\kappa_2 = 1 - \prod_{i=2}^{r} \frac{p-k_i}{p-1}$ *and* $(k_2, k_3, k_4, \ldots, k_r) = (7, 2, 8, \ldots, 8)$.

Proof. Given an adversary against FS-WUR of FS-BP with respect to a canonical NIZK simulator $\mathcal{S}_{\mathsf{FS\text{-}BP}}$, $\mathcal{A}_{ur} = (\mathcal{A}_1, \mathcal{A}_2)$ (Definition 3.3), our ultimate goal is to construct an adversary $\mathcal{B}$ that breaks the discrete-log relation. First, consider the following hybrid games.

– $\mathsf{G}_0(\mathcal{A}_{ur})$: This is identical to the game defining the FS-WUR advantage, except that G_0 uses an alternative simulator $\mathcal{S}'_{\mathsf{FS\text{-}BP}}$ instead of $\mathcal{S}_{\mathsf{FS\text{-}BP}}$. That is, an adversary $\mathcal{A}_{ur}$ is given access to $(\mathcal{S}_1, \mathcal{S}'_2)$ and the game outputs 1 if $\mathcal{A}_{ur}$ outputs a valid proof $\mathcal{T}$ that shares prefix with simulated proof $\widetilde{\mathcal{T}}$ output by $\mathcal{S}_1$, where $(\mathcal{S}_1, \mathcal{S}'_2)$ are now the wrappers defined for $\mathcal{S}'_{\mathsf{FS\text{-}BP}}$. Since both $\mathcal{S}_{\mathsf{FS\text{-}BP}}$ and $\mathcal{S}'_{\mathsf{FS\text{-}BP}}$ are canonical NIZK simulators derived from perfect HVZK simulators, the view of $\mathcal{A}_{ur}$ is identical in both games. Hence, we have

$$\Pr[\mathsf{G}_0(\mathcal{A}_{ur}) = 1] = \mathbf{Adv}^{\mathsf{FS\text{-}WUR}}_{\mathsf{FS\text{-}BP}}(\mathcal{A}_{ur}, \mathcal{S}_{\mathsf{FS\text{-}BP}}).$$

– $\mathsf{G}_1(\mathcal{A}_{ur})$: This is identical to G_0, except that the game aborts by setting the flag *bad* if $\mathcal{T} = (a_1, c_1, \ldots, c_r, a_{r+1})$ output by $\mathcal{A}_{ur}$ is constructed without querying all r random oracles while meeting the winning condition of G_0, i.e., G_1 sets $\mathit{bad} = \mathsf{true}$ if $\exists i \in [1, r]$ such that $\mathcal{Q}_{1,i}[\mathsf{pp}, x, a_1, c_1, \ldots, a_i] = \bot$ $\wedge\ \mathcal{V}^{\mathsf{H}}_{\mathsf{FS}}(\mathsf{pp}, x, \mathcal{T}) = 1 \wedge \mathcal{T}$ shares prefix with $\widetilde{\mathcal{T}}$. Since $\mathcal{T}$ must be accepted by $\mathcal{V}_{\mathsf{FS}}$, the *bad* flag is set only if the random oracle output exactly matches the challenge guessed by $\mathcal{A}_{ur}$. This happens with probability at most $1/(p-1)$ and we obtain

$$\Pr[\mathsf{G}_0(\mathcal{A}_{ur}) = 1] \leq \Pr[\mathsf{G}_1(\mathcal{A}_{ur}) = 1] + 1/(p-1).$$

Note that, this is assuming the adversary guesses only one challenge, as the advantage in this case is higher than guessing multiple challenges.

In the rest of the proof, we bound the probability $\Pr[\mathsf{G}_1(\mathcal{A}_{ur}) = 1]$. We do so by showing a reduction $\mathcal{B}$ that, given an adversary $\mathcal{A}_{ur} = (\mathcal{A}_1, \mathcal{A}_2)$ emulates its view in G_1 and finds a non-trivial DL-REL relation between $\bar{\mathbf{g}} = (g, h, u, \mathbf{g}, \mathbf{h}) \in \mathbb{G}^{2n+3}$ by obtaining a tree-of-transcript. Concrete description of $\mathcal{B}$ is provided in Alg. 5. To invoke the result of [AFK21], we introduce a wrapper algorithm $\mathcal{A}'$ for the unique response adversary analogously to the one for a knowledge soundness adversary from Section 2.8. $\mathcal{A}'$ takes $(x, \widetilde{\mathcal{T}}, \mathsf{st})$ as input, forwards $(\widetilde{\mathcal{T}}, \mathsf{st})$ to $\mathcal{A}_2$ and relays all the random oracle queries made by $\mathcal{A}_2$ to $\mathcal{S}_1$. When $\mathcal{A}_2$ outputs a transcript $\mathcal{T}$, $\mathcal{A}'$ returns the postprocessed output $(\mathbf{I}, \mathcal{T}, b, i^*)$, where $I_1 = (x, a_1), \ldots, I_r = (x, a_1, \ldots, a_r), \mathbf{I} = (I_1, \ldots, I_r)$,

$$b = \left(\forall i \in [1, r] : \mathcal{Q}_{1,i}[\mathsf{pp}, x, a_1, c_1, \ldots, a_i] \neq \bot\ \wedge\ \mathcal{V}^{\mathcal{S}_1}_{\mathsf{FS}}(\mathsf{pp}, x, \mathcal{T})\ \wedge\ \exists i^* \in [1, r] : (\mathcal{T}|_{i^*} = \widetilde{\mathcal{T}}|_{i^*}\ \wedge\ \mathcal{T} \neq \widetilde{\mathcal{T}})\right)$$

and $i^* := 0$ if $b = 0$. Note that the bit b indicates whether $\mathcal{A}_{ur}$ would win G_1 or not. Since this is merely a syntactic change, the success probability of $\mathcal{A}'$ is

$$\epsilon(\mathcal{A}') := \Pr\left[b = 1\ :\ \begin{array}{l}(\mathsf{pp}, \mathsf{H}) \leftarrow \mathsf{Setup}_{\mathsf{FS}}(1^\lambda); (x, \mathsf{st}) \leftarrow \mathcal{A}_1^{\mathcal{S}_1}(\mathsf{pp});\\ \widetilde{\mathcal{T}} \leftarrow \mathcal{S}'_2(x); (\mathbf{I}, \mathcal{T}, b, i^*) \leftarrow \mathcal{A}'^{\mathcal{S}_1}(x, \widetilde{\mathcal{T}}, \mathsf{st});\end{array}\right] = \Pr[\mathsf{G}_1(\mathcal{A}_{ur}) = 1].$$

In what follows, we explain how to build a (sub)tree of accepting transcripts (Definition 2.9) from which $\mathcal{B}$ can derive a non-trivial discrete log relation.

Obtaining tree-of-transcript. Suppose after making $q + r$ queries to the random oracle $\mathcal{A}'$ generates a winning transcript $\mathcal{T}$ which differs with simulated $\widetilde{\mathcal{T}}$ for the first time at the $(i^* + 1)$-th prover message.

For relation $\mathcal{R}_{\mathsf{ACSPf}}$, statement x consists solely of elements in $\mathbb{Z}_p$. Therefore $\mathcal{B}$ knows all representation for elements in $\widetilde{\mathcal{T}}$ in terms of an equivalent representation purely in terms of $(\mathbf{g}, \mathbf{h}, g, h, u)$. Our goal is now to obtain the same for $\mathcal{T}$.

Recall, interactive BP is a $(k_{1,0}, k_{1,1}, \ldots, k_r)$-special-sound public coin interactive protocol, where $k_{1,0} = n, k_{1,1} = (Q+1), k_2 = 7, k_3 = 2$ and $k_4 = 8, \ldots, k_r = 8$.[14] Thus, the Fiat-Shamir transformation of BP is knowledge sound (Corollary 2.1). In particular, there exists a series of sub-extractors $(\mathcal{E}_1, \ldots, \mathcal{E}_r)$ (defined in Fig. 1), where each sub-extractor $\mathcal{E}_\ell$ halts in expected polynomial time in λ and q and outputs a $(1, \ldots, 1, k_\ell, \ldots, k_r)$-tree of transcript with probability at least $\frac{\epsilon(\mathcal{A})-(q+1)\cdot\kappa_\ell}{1-\kappa_\ell}$ (see Lemma 2.1).

To obtain group representation for $\mathcal{T}$, $\mathcal{B}$ will attempt to build a $(1, \ldots, 1, k_{i^*+1}, \ldots, k_r)$-tree by running a slightly modified extractor $\mathcal{E}'_1$, which is again composed of recursive calls to sub-extractors defined in Fig. 6. We define the base case extractor $\mathcal{E}'_{r+1} := \mathcal{A}'$ as described above and $\mathcal{E}'_\ell$ for $\ell \in [1, r]$ is defined in a similar way as $\mathcal{E}_\ell$ with the following differences.

- $\mathcal{E}'_\ell$ gets black-box access to (the second stage of) weak unique response adversary $\mathcal{A}_2$ via $\mathcal{A}'$.
- Each $\mathcal{E}'_\ell$, including $\mathcal{E}'_1$, relays all random oracle queries made in the first run outside to $\mathcal{B}$ who answers these to be consistent with the challenges in the simulated transcript, and with previously answered queries. New queries unrelated to $\widetilde{\mathcal{T}}$ are answered by lazy sampling. The random oracle queries made in the repeat loop are answered locally by each sub-extractor as before.
- To allow the base case extractor to postprocess the transcript with prefix check, $\mathcal{E}'_\ell$ gets an additional input $\widetilde{\mathcal{T}}$. If $\mathcal{T}$ output during the initial run of $\mathcal{A}'$ has no shared prefix with $\widetilde{\mathcal{T}}$, there is no need to build a tree of transcript and it simply aborts.
- Each sub-extractor executes the first run step completely but the repeat loop step only partially. After the first run of $\mathcal{A}'$, it outputs $\mathcal{T}$ such that it differs from $\widetilde{\mathcal{T}}$ for the first time at the (i^*+1)-th prover message. The repeat loop for sub-extractors $\mathcal{E}'_\ell$, for $\ell \in [i^*+1, r]$, is the same as before, i.e., they output a $(1, \ldots, 1, k_i, \ldots, k_r)$-tree. On the other hand, sub-extractors $\mathcal{E}'_\ell$, for $\ell \in [1, i^*]$, do not run the repeat loop. Instead they merely propagate the last tree that was build, i.e., the $(1, \ldots, 1, k_{i^*+1}, \ldots, k_r)$-tree output by $\mathcal{E}'_{i^*+1}$. Finally, $\mathcal{E}'_1$ outputs a $(1, \ldots, 1, k_{i^*+1}, \ldots, k_r)$-tree. This is done to avoid rewinding $\mathcal{A}$ before the (i^*+1)-th round message, which as mentioned before, is necessary to maintain indistinguishability of $\mathcal{A}_2$'s view from the view in $\mathsf{FS\text{-}WUR}$ experiment.

$\mathcal{E}'_1$ either outputs abort or outputs a $(1, \ldots, k_{i^*+1}, \ldots, k_r)$-tree called tr. In particular, the output of $\mathcal{E}'_1$ is a tree that has transcripts sharing common prefix with $\mathcal{T}$ up until the (i^*+1)-th prover message, after which, it forks. Using tr, $\mathcal{A}$ derives representation for $\mathcal{T}$ for group elements starting from round i^*+1. Given these representations, $\mathcal{B}$ can break discrete-log.

Cases by case analysis. Note that, the very first time $\mathcal{A}'$ is invoked by $\mathcal{B}$, there has been no rewinding till now. Thus, this execution is identical to G_1 (which is identical to G_0 and $\mathsf{FS\text{-}WUR}$ up to the negligible bad event). Since $\mathcal{A}_{ur}$ wins the $\mathsf{FS\text{-}WUR}$ game, $\mathcal{T}$ (output in game G_1) is an accepting transcript for statement x which is different from $\widetilde{\mathcal{T}}$, but has a common prefix. Therefore, at least the first two messages must be equal. In particular, $\mathcal{S}'_{\mathsf{FS\text{-}BP}}$ outputs transcript of the form:

$$\widetilde{\mathcal{T}} = (\tilde{A}_I, \tilde{A}_O, \tilde{S}; \tilde{y}, \tilde{z}; (\tilde{T}_i)_{i\in\mathscr{S}}; \tilde{x}; \tilde{\beta}_x, \tilde{\mu}, \tilde{\hat{t}}, \tilde{w}, \tilde{L}_1, \tilde{R}_1, \tilde{x}_1, \ldots, \tilde{L}_m, \tilde{R}_m, \tilde{x}_m, \tilde{a}, \tilde{b})$$

and $\mathcal{A}_{ur}$ outputs transcripts of the form:

$$\mathcal{T} = (\tilde{A}_I, \tilde{A}_O, \tilde{S}; \tilde{y}, \tilde{z}; (T_i)_{i\in\mathscr{S}}; x; \beta_x, \mu, \hat{t}, w, L_1, R_1, x_1, \ldots, L_m, R_m, x_m, a, b).$$

We now proceed with a case by case analysis based on the first message in $\mathcal{T}$ which is different from $\widetilde{\mathcal{T}}$.

If $\tilde{T}_i \neq T_i$ for some $i \in \mathscr{S}$ (i.e. $i^* = 1$), then the verification equation satisfied by $\mathcal{T}$ is:

$$(\mathbf{g}^{(m)})^a (\mathbf{h}^{(m)})^b (u')^{ab} =$$
$$\left(\prod_{i=1}^m L_i^{x_i^2}\right) \cdot h^{-\mu} \cdot \tilde{A}_I^x \cdot \tilde{A}_O^{x^2} \cdot (\tilde{\mathbf{h}}')^{-\tilde{\mathbf{y}}^n} \cdot \tilde{W}_L^x \cdot \tilde{W}_R^x \cdot \tilde{W}_O \cdot \tilde{S}^{x^3} \cdot (u')^{\hat{t}} \cdot \left(\prod_{i=1}^m R_i^{x_i^{-2}}\right).$$

[14] Since the extractor of the inner product argument requires four challenges x_1, x_2, x_3, x_4 such that $x_i \neq \pm x_j$ for $1 \leq i < j \leq 4$, at most 8 distinct challenges suffice for meeting this condition.

Algorithm 5: $\mathcal{B}(\bar{\mathbf{g}})$

1. Using a received discrete log instance $\bar{\mathbf{g}} \in \mathbb{G}^{2n+3}$, construct $\mathsf{pp} = (n, Q, g, h, u, \mathbf{g}, \mathbf{h})$. Initialize $\mathcal{S}'_{\mathsf{FS\text{-}BP}}$ with pp and run $\mathcal{A}_1(\mathsf{pp})$ to receive a statement x, where $\mathcal{B}$ answers $\mathcal{A}_1$'s random oracle queries and maintains the query tables $\mathcal{Q}$ as $\mathcal{S}_1$ would.
2. Follow the procedures of $\mathcal{S}_2(x)$ to generate a simulated transcript $\widetilde{\mathcal{T}}$ and update $\mathcal{Q}$. Note that by examining the internal operations of $\mathcal{S}'_{\mathsf{FS\text{-}BP}}$, $\mathcal{B}$ can obtain the group representation of $\widetilde{\mathcal{T}}$ w.r.t. input generators.
3. Run $\mathcal{E}'_1(x, \widetilde{\mathcal{T}}, \mathsf{st})$ as follows to obtain output $(\mathbf{I}, \mathsf{tr}, b, i^*)$: If the random oracle query has been asked before, answer with the recorded reply. Answer all new queries by lazy sampling. In addition, record all random oracle responses in $\mathcal{Q}$.
4. If $b = 0$, abort. Else, let $\mathcal{T}$ be a transcript derived from the path in tr corresponding to the initial run of $\mathcal{A}'$. tr corresponds to the $(1, \ldots, 1, k_{i^*+1}, \ldots, k_r)$-tree of transcripts. Output discrete-log relation by case-by-case analysis.

Algorithm 6: $\mathcal{E}'_\ell(x, \widetilde{\mathcal{T}}, \mathsf{st})$

1. $\mathsf{First\ Run}$: Run $\mathcal{E}'_{\ell+1}(x, \widetilde{\mathcal{T}}, \mathsf{st})$ as follows to obtain $(\mathbf{I}, \mathsf{tr}_1, b, i^*)$.
   (a) Relay all the random oracle queries made by $\mathcal{E}'_{\ell+1}$ externally to $\mathcal{E}'_{\ell-1}$ and record query response pair.
   (b) Set $j = I_\ell$ and let c_j be the response to query j.
2. If $b = 0$, abort with output $(\mathbf{I}, \perp, 0, 0)$.
3. Else, if $\ell \leq i^*$, output $(\mathbf{I}, \mathsf{tr}_1, b, i^*)$.
4. $\mathsf{Repeat\ Run}$: Else, initialize challenge set $\mathsf{Ch}' = \mathsf{Ch} \setminus \{c_j\}$ and repeat:
   (a) If $\mathsf{Ch}' = \varnothing$ goto Step 5.
   (b) Sample $c'_j \in \mathsf{Ch}'$ and update $\mathsf{Ch}' := \mathsf{Ch}' \setminus \{c'_j\}$.
   (c) Run $\mathcal{E}'_{\ell+1}(x, \widetilde{\mathcal{T}}, \mathsf{st})$ as follows to obtain $(\hat{\mathbf{I}}, \hat{\mathsf{tr}}, \hat{b}, \hat{i}^*)$, aborting right after the initial run of $\mathcal{A}'$ if $\hat{I}_\ell \neq I_\ell$: answer the random oracle query to j with c'_j, while answering all other queries consistently if the query was performed by $\mathcal{E}'_{\ell+1}$ already on a previous run and with a fresh random value in Ch otherwise.
   (d) If $k_\ell - 1$ additional challenges c'_j with $b' = 1$ and $I'_\ell = I_\ell$ have been found then goto Step 6.
5. Output $(\mathbf{I}, \perp, 0, 0)$.
6. Output $(\mathbf{I}, (\mathsf{tr}_1, \ldots, \mathsf{tr}_{k_\ell}), 1, i^*)$, where $\mathsf{tr}_1, \ldots, \mathsf{tr}_{k_\ell}$ are the k_ℓ accepting $(1, \ldots, 1, k_{\ell+1}, \ldots, k_r)$-trees.

(The values $\widetilde{W}_{(\cdot)}$ and $\tilde{\mathbf{h}}'$ are also marked as $\tilde{(\cdot)}$ to remind the reader that they are the same in both $\mathcal{T}$ and $\widetilde{\mathcal{T}}$. Remember that $\mathbf{g}^{(m)}, \mathbf{h}^{(m)}$ are different in the two transcripts and they are generated as part of the InPrd). Dividing it by the verification equation for the simulated transcript, we get

$$(\mathbf{g}^{(m)})^a(\mathbf{h}^{(m)})^b(u')^{ab} \cdot (\tilde{\mathbf{g}}^{(m)})^{-\tilde{a}}(\tilde{\mathbf{h}}^{(m)})^{-\tilde{b}}(\tilde{u}')^{-\tilde{a}\tilde{b}} = \tag{4}$$

$$\left(\prod_{i=1}^{m} L_i^{x_i^2}\right)\left(\prod_{i=1}^{m} \tilde{L}_i^{-\tilde{x}_i^2}\right)\left(\prod_{i=1}^{m} R_i^{x_i^{-2}}\right)\left(\prod_{i=1}^{m} \tilde{R}_i^{-\tilde{x}_i^{-2}}\right)$$

$$\cdot\, h^{-(\mu-\tilde{\mu})} \cdot \tilde{A}_I^{x-\tilde{x}} \cdot \tilde{A}_O^{x^2-\tilde{x}^2} \cdot \widetilde{W}_L^{x-\tilde{x}} \cdot \widetilde{W}_R^{x-\tilde{x}} \cdot \tilde{S}^{x^3-\tilde{x}^3} \cdot (u')^{\hat{t}} \cdot (\tilde{u}')^{-\hat{t}}. \tag{5}$$

We rearrange the exponents w.r.t. the generators $(g, h, \mathbf{g}, \mathbf{h}, u)$. We will focus on the exponent of g. Note that, from simulator $\mathcal{S}'_{\mathsf{FS\text{-}BP}}$, $\mathcal{B}$ knows the representation for $\widetilde{\mathcal{T}}$. Now, we derive representation for $\mathcal{T}$. Looking ahead, we will aim to derive representation of L_i and R_i without using g. On the other hand, exponent of g in $\widetilde{\mathcal{T}}$ will be non-zero with high probability. This will be crucial for breaking DL-REL using the isolated exponent of g.
Recall, $\mathcal{B}$ obtains tr with $i^* = 1$ from executing $\mathcal{E}'_1$, where tr is a $(1, 7, 2, 8, \ldots, 8)$-tree of transcript. Given the tree tr, we will derive the representation for all L_i, R_i terms in Eq. (5) recursively, starting from L_m, R_m. Consider the subtrees in tr rooted at the last level (the $r = (m+3)$-th level). In the following we use $\hat{L}_m, \hat{R}_m$ to generically denote the root for all these subtrees, although note that, these most likely take different values in each subtree. For e.g., for the very first subtree $\hat{L}_m = L_m$ and $\hat{R}_m = R_m$. The last round message in each transcript is two field elements generically denoted by $\hat{a}, \hat{b}$. Again, note that for the first transcript in the first subtree, $\hat{a} = a, \hat{b} = b$. Finally, we denote the last round challenge in each of the transcript generically by $\hat{x}_m$, the group parameters by $\hat{g}$ and $\hat{h}$, and the statement derived in the previous round by $\hat{P}_{m-1}$. Each subtree rooted at the r-th level forks at most 8 times with respect to four distinct challenges $\hat{x}_{m,1}, \ldots, \hat{x}_{m,4}$, such that $\hat{x}_{m,i} \neq \pm\hat{x}_{m,j}$ for $i \neq j$. Using the first three challenges, one can derive a group representation for $\hat{L}_m, \hat{R}_m$ in an analogous manner to Theorem 1 of [BBB+18]. Since all these transcripts are valid, for $\hat{x}_m \in \{\hat{x}_{m,1}, \ldots, \hat{x}_{m,4}\}$, they pass the verification check,

$$\hat{L}_m^{\hat{x}_m^2} \cdot \hat{P}_{m-1} \cdot \hat{R}_m^{\hat{x}_m^{-2}} = (\hat{g})^{\hat{a}} \cdot \left(\hat{h}\right)^{\hat{b}} \cdot u^{\langle \hat{a}, \hat{b} \rangle} \tag{6}$$

Rewriting (6) in terms of $\hat{\mathbf{g}}^{(m-1)}$ and $\hat{\mathbf{h}}^{(m-1)}$ (as defined by honest verifier's algorithm in Protocol 2, given partial transcript till round $m-2$.), we get,

$$\hat{L}_m^{\hat{x}_m^2} \cdot \hat{P}_{m-1} \cdot \hat{R}_m^{\hat{x}_m^{-2}} = \left(\left(\hat{\mathbf{g}}_{[:n_m]}^{(m-1)}\right)^{\hat{x}_m^{-1}} \circ \left(\hat{\mathbf{g}}_{[n_m:]}^{(m-1)}\right)^{\hat{x}_m}\right)^{\hat{a}} \cdot \left(\left(\hat{\mathbf{h}}_{[:n_m]}^{(m-1)}\right)^{\hat{x}_m} \circ \left(\hat{\mathbf{h}}_{[n_m:]}^{(m-1)}\right)^{\hat{x}_m^{-1}}\right)^{\hat{b}} \cdot u^{\langle \hat{a}, \hat{b} \rangle} \tag{7}$$

Using the first three challenges, compute ν_1, ν_2, ν_3 such that,

$$\sum_{j=1}^{3} \nu_j \cdot \hat{x}_{m,j}^2 = 1, \quad \sum_{j=1}^{3} \nu_j = 0 \quad \sum_{j=1}^{3} \nu_j \cdot \hat{x}_{m,j}^{-2} = 0 \tag{8}$$

Now taking linear combination of three relations in (7), wrt to coefficients computed in (8), we get $\hat{\mathbf{a}}_L^{(m-1)}\ \hat{\mathbf{b}}_L^{(m-1)}, \hat{c}_L^{(m-1)}$ such that $\hat{L}_m = \left(\hat{\mathbf{g}}^{(m-1)}\right)^{\hat{\mathbf{a}}_L^{(m-1)}} \cdot \left(\hat{\mathbf{h}}^{(m-1)}\right)^{\hat{\mathbf{b}}_L^{(m-1)}} \cdot u^{\hat{c}_L^{(m-1)}}$. Solving lhs in Eq. (8) for rhs evaluations $(0, 1, 0)$ and $(0, 0, 1)$ gives us coefficients for $\hat{P}_{m-1}$ and $\hat{R}_m$, resp. Moreover, using the forth challenge, one can argue that it must be the case that,

$$\begin{gathered}
\hat{\mathbf{a}}_{L,[:n_m]}^{(m-1)} = 0, \quad \hat{\mathbf{a}}_{R,[n_m:]}^{(m-1)} = 0, \quad \hat{\mathbf{b}}_{R,[:n_m]}^{(m-1)} = 0, \quad \hat{\mathbf{b}}_{L,[n_m:]}^{(m-1)} = 0 \\
\hat{\mathbf{a}}_{L,[n_m:]}^{(m-1)} = \hat{\mathbf{a}}_{P,[:n_m]}^{(m-1)}, \quad \hat{\mathbf{a}}_{R,[:n_m]}^{(m-1)} = \hat{\mathbf{a}}_{P,[n_m:]}^{(m-1)} \\
\hat{\mathbf{b}}_{L,[:n_m]}^{(m-1)} = \hat{\mathbf{b}}_{P,[n_m:]}^{(m-1)}, \quad \hat{\mathbf{b}}_{R,[n_m:]}^{(m-1)} = \hat{\mathbf{b}}_{P,[:n_m]}^{(m-1)} \\
\text{and } \left\langle \hat{\mathbf{a}}_{P,[:n_m]}^{(m-1)}, \hat{\mathbf{b}}_{P,[n_m:]}^{(m-1)} \right\rangle = \hat{c}_L^{(m-1)}, \quad \left\langle \hat{\mathbf{a}}_{P,[n_m:]}^{(m-1)}, \hat{\mathbf{b}}_{P,[:n_m]}^{(m-1)} \right\rangle = \hat{c}_R^{(m-1)}
\end{gathered} \tag{9}$$

We skip the exact argument to derive conditions in Eq. (9) as it follows directly from the extraction analysis of BP in [BBB+18]. This procedure is replicated to obtain group representation for $\hat{L}_{m-1}, \hat{R}_{m-1}, \hat{P}_{m-2}$, given $\hat{\mathbf{a}}_P^{(m-1)}$ and $\hat{\mathbf{b}}_P^{(m-1)}$, and, in general for $\hat{L}_i, \hat{R}_i, \hat{P}_{i-1}$, given the $\hat{\mathbf{a}}_P^{(i)}$ and $\hat{\mathbf{b}}_P^{(i)}$, and using the following check on four different challenges $\hat{x}_i$,

$$\hat{L}_i^{\hat{x}_i^2} \cdot \hat{P}_{i-1} \cdot \hat{R}_i^{\hat{x}_i^{-2}} = \left(\left(\hat{\mathbf{g}}_{[:n_i]}^{(i-1)}\right)^{\hat{x}_i^{-1}} \circ \left(\hat{\mathbf{g}}_{[n_i:]}^{(i-1)}\right)^{\hat{x}_i}\right)^{\hat{\mathbf{a}}_P^{(i)}} \cdot \left(\left(\hat{\mathbf{h}}_{[:n_i]}^{(i-1)}\right)^{\hat{x}_i} \circ \left(\hat{\mathbf{h}}_{[n_i:]}^{(i-1)}\right)^{\hat{x}_i^{-1}}\right)^{\hat{\mathbf{b}}_P^{(i)}} \cdot u^{\left\langle \hat{\mathbf{a}}_P^{(i)}, \hat{\mathbf{b}}_P^{(i)} \right\rangle}$$

Once all the group representations have been derived for transcript $\mathcal{T}$, substitute these values in the verification equation in Eq. (5), and equate representation for g on lhs and rhs. Notice that the g-component for L_i (resp. R_i) is 0. The only elements with a non-zero component for g are: the simulated $\tilde{A}_I$ and $\tilde{S}$ that have ρ_I and $-\rho_I \tilde{x}^{-2}$ in the exponents of g, respectively.

Then the exponent of g in (5) is

$$-\rho_I \tilde{x}^{-2} x^3 + \rho_I x \tag{10}$$

Consider one other subtree rooted at level 3, i.e., it has the same messages T_i's as in $\mathcal{T}$ and then fork for a different challenge $\hat{x}$. Now consider one transcript from this subtree. Since this is an accepting transcript as well, we can derive group representation for $\hat{L}_i, \hat{R}_i$ for it. Once again, the coefficient for g and h will come out to be zero. Equating exponents of g in (5), for challenge $\hat{x}$, we get,

$$-\rho_I \tilde{x}^{-2} \hat{x}^3 + \rho_I \hat{x} \tag{11}$$

If either (10) or (11) is non-zero then we find a non-trivial DL solution, since the left-hand side of Eq. (5) has g-component 0. Both (10) and (11) cannot be 0 at the same time as this would imply that polynomial $-\rho_I \tilde{x}^{-2} X^3 + \rho_I X$ is a zero-polynomial, and so $\rho_I = 0$. However, $\rho_I = 0$ happens with probability $1/(p)$ because it is chosen uniformly by the simulator.

If $\beta_x \neq \tilde{\beta}_x$ or $\hat{t} \neq \tilde{\hat{t}}$ (i.e. $i^* = 2$), then we have another transcript

$$\mathcal{T} = (\tilde{A}_I, \tilde{A}_O, \tilde{S}; \tilde{y}, \tilde{z}; (\tilde{T}_i)_{i \in \mathscr{S}}; \tilde{x}, \beta_x, \mu, \hat{t}, w, L_1, R_1, x_1, \ldots, L_m, R_m, x_m, a, b).$$

Since both simulated and adversarial transcripts satisfy the verification equation w.r.t. the same R, we have

$$g^{\hat{t}} h^{\beta_x} = R = g^{\tilde{\hat{t}}} h^{\tilde{\beta}_x}$$

which leads to a non-trivial DL relation.

If $\mu \neq \tilde{\mu}$ (i.e. $i^* = 2$), the analysis is similar to the case where $\tilde{T}_i \neq T_i$. The verification equation satisfied by $\mathcal{T}$ is

$$\begin{aligned}&(\mathbf{g}^{(m)})^a (\mathbf{h}^{(m)})^b (u')^{ab} \\ &= \left(\prod_{i=1}^m L_i^{x_i^2}\right) \cdot h^{-\mu} \cdot \tilde{A}_I^{\tilde{x}} \cdot \tilde{A}_O^{\tilde{x}^2} \cdot (\tilde{\mathbf{h}}')^{-\tilde{\mathbf{y}}^n} \cdot \tilde{W}_L^{\tilde{x}} \cdot \tilde{W}_R^{\tilde{x}} \cdot \tilde{W}_O \cdot \tilde{S}^{\tilde{x}^3} \cdot (u')^{\hat{t}} \cdot \left(\prod_{i=1}^m R_i^{x_i^{-2}}\right).\end{aligned}$$

Dividing it by the verification equation for the simulated transcript, we get

$$\begin{aligned}&(\mathbf{g}^{(m)})^a (\mathbf{h}^{(m)})^b (u')^{ab} \cdot (\tilde{\mathbf{g}}^{(m)})^{-\tilde{a}} (\tilde{\mathbf{h}}^{(m)})^{-\tilde{b}} (\tilde{u}')^{-\tilde{a}\tilde{b}} \\ &= \left(\prod_{i=1}^m L_i^{x_i^2}\right)\left(\prod_{i=1}^m \tilde{L}_i^{-\tilde{x}_i^2}\right)\left(\prod_{i=1}^m R_i^{x_i^{-2}}\right)\left(\prod_{i=1}^m \tilde{R}_i^{-\tilde{x}_i^{-2}}\right) \\ &\cdot h^{-(\mu - \tilde{\mu})} \cdot (u')^{\hat{t}} \cdot (\tilde{u}')^{-\hat{t}}.\end{aligned} \tag{12}$$

We rearrange the exponents w.r.t. the generators $(g, h, \mathbf{g}, \mathbf{h}, u)$. Let us focus on the exponent of h. Again, as explained in the case for $T_i \neq \tilde{T}_i$,we derive group representation for L_i, R_i terms in (12) given a $(1, 1, 2, 8, \ldots, 8)$-tree tr with $i^* = 2$, and observe that the h-component for L_i, R_i is 0. Then the exponent of h in (12) is $-(\mu - \tilde{\mu})$. Using the same argument as before, since the h-component in the left-handside of 12 is 0, if $\mu \neq \tilde{\mu}$ we obtain non-trivial DL relation. Otherwise, $\mu = \tilde{\mu}$, which happens with probability $1/p$ since the simulator picks $\tilde{\mu}$ uniformly at random.

If $L_i \neq \tilde{L}_i$ or $R_i \neq \tilde{R}_i$ for some $i \in [1, m]$ (i.e. $i^* = i + 2$), $\mathcal{B}$ obtains a $(1, \ldots, 1, 8, \ldots, 8)$-tree tr after executing $\mathcal{E}_1'$. tr is such that all the transcripts share the same messages $(A_I, A_O, \ldots, L_i, R_i)$ and then fork with respect to challenges x_i. Using the same procedure as discussed in the case of $T_i \neq \tilde{T}_i$,

$\mathcal{B}$ can derive the group representation for L_i, R_i and for P_{i-1}. Let the derived representation for P_{i-1} in terms of group parameters $\mathbf{g}^{(i-1)}, \mathbf{h}^{(i-1)}$ be $\mathbf{a}_P^{(i-1)}, \mathbf{b}_P^{(i-1)}, c_P^{(i-1)}$, and the representation submitted by the simulator for $\tilde{P}_{i-1}$ be $\tilde{\mathbf{a}}^{(i-1)}, \tilde{\mathbf{b}}^{(i-1)}, \tilde{c}^{(i-1)}$. Note that, $P_{i-1} = \tilde{P}_{i-1}$, and $\mathbf{g}^{(i-1)}, \mathbf{h}^{(i-1)}$ are the same in both the transcripts since they deviate for the first time at $(i+3)$-th prover round.

$$\begin{aligned} \tilde{P}_{i-1} = P_{i-1} &= \left(\mathbf{g}^{(i-1)}\right)^{\mathbf{a}_P^{(i-1)}} \cdot \left(\mathbf{h}^{(i-1)}\right)^{\mathbf{b}_P^{(i-1)}} \cdot u^{c_P^{(i-1)}} \\ \text{and } &= \left(\mathbf{g}^{(i-1)}\right)^{\tilde{\mathbf{a}}^{(i-1)}} \cdot \left(\mathbf{h}^{(i-1)}\right)^{\tilde{\mathbf{b}}^{(i-1)}} \cdot u^{\tilde{c}^{(i-1)}} \end{aligned} \tag{13}$$

The exponent in (13) must match otherwise we obtain non-trivial DL relation.

Similar to the conditions in (9), from the analysis in [BBB+18], we also know that the derived $\mathbf{a}_L^{(i-1)}, \mathbf{b}_L^{(i-1)}, \ldots$ must satisfy the following,

$$\begin{gathered} \mathbf{a}_{L,[:n_i]}^{(i-1)} = 0, \quad \mathbf{a}_{R,[n_i:]}^{(i-1)} = 0, \quad \mathbf{b}_{R,[:n_i]}^{(i-1)} = 0, \quad \mathbf{b}_{L,[n_i:]}^{(i-1)} = 0 \\ \mathbf{a}_{L,[n_i:]}^{(i-1)} = \tilde{\mathbf{a}}_{[:n_i]}^{(i-1)}, \quad \mathbf{a}_{R,[:n_i]}^{(i-1)} = \tilde{\mathbf{a}}_{[n_i:]}^{(i-1)} \\ \mathbf{b}_{L,[:n_i]}^{(i-1)} = \tilde{\mathbf{b}}_{[n_i:]}^{(i-1)}, \quad \mathbf{b}_{R,[n_i:]}^{(i-1)} = \tilde{\mathbf{b}}_{[:n_i]}^{(i-1)}, \\ \text{and } \left\langle \tilde{\mathbf{a}}_{[:n_i]}^{(i-1)}, \tilde{\mathbf{b}}_{[n_m:]}^{(i-1)} \right\rangle = \tilde{c}_L^{(i-1)}, \quad \text{and } \left\langle \tilde{\mathbf{a}}_{[n_i:]}^{(i-1)}, \tilde{\mathbf{b}}_{[:n_i]}^{(i-1)} \right\rangle = \tilde{c}_R^{(i-1)} \end{gathered} \tag{14}$$

Substituting these values for L_i and (and similarly for R_i), we get,

$$L_i = \left(\mathbf{g}_{[n_i:]}^{(i-1)}\right)^{\tilde{\mathbf{a}}_{[:n_i]}^{(i-1)}} \left(\mathbf{h}_{[:n_i]}^{(i-1)}\right)^{\tilde{\mathbf{b}}_{[n_i:]}^{(i-1)}} u^{\left\langle \tilde{\mathbf{a}}_{[:n_i]}^{(i-1)}, \tilde{\mathbf{b}}_{[n_m:]}^{(i-1)} \right\rangle}$$

which is exactly how $\tilde{L}_i$ (and $\tilde{R}_i$) is computed by the simulator. Thus, $L_i = \tilde{L}_i$ and $R_i = \tilde{R}_i$.

If $a \neq \tilde{a}$ or $b \neq \tilde{b}$ (i.e. $i^* = r$), Since $\mathcal{T}$ is accepting,

$$P_{\log(n)} = g^a h^b u^{ab}$$

Similarly, for $\tilde{\mathcal{T}}$,

$$P_{\log(n)} = g^{\tilde{a}} h^{\tilde{b}} u^{\tilde{a}\tilde{b}}$$

Note that the statement $P_{\log(n)}$ and parameters g, h with be the same for both transcripts since they are defined using previous round messages. Dividing the two equation, $\mathcal{B}$ obtains a non-trivial DL-REL between g, h, u, since either $a \neq \tilde{a}$ or $b \neq \tilde{b}$.

$$g^{a-\tilde{a}} h^{b-\tilde{b}} u^{ab-\tilde{a}\tilde{b}} = 1_{\mathbb{G}}$$

Recall that g and h are constructed by taking linear combinations of $(g_1, \ldots, g_n)$ and $(h_1, \ldots, h_n)$ with non-zero coefficients, respectively. Therefore, a non-trivial discrete log relation between g, h, u can be converted into the one between $\mathbf{g}, \mathbf{h}, u$.

Concrete advantage of the adversary. If $\mathcal{A}_{ur}$ succeeds in the G_1, and the execution of $\mathcal{E}_1'$ outputs an appropriate tree of transcript, then $\mathcal{B}$ succeeds in deriving a non-trivial DL relation. Let ℓ^* be the first round where the prover's message deviates from the simulated one, i.e., $\ell^* = i^* + 1$. Let $K_{\ell^*} = \prod_{i=\ell^*}^{r} k_i$. From Lemma 2.1, the extractor $\mathcal{E}_1'$ makes expected number of at most $K_{\ell^*} + q \cdot (K_{\ell^*} - 1)$ queries to $\mathcal{A}'$ and outputs tuple containing a $(1, \ldots, 1, k_{\ell^*}, \ldots, k_r)$-tree, and $b = 1$ with probability at least

$$\frac{\epsilon(\mathcal{A}') - (q+1) \cdot \kappa_{\ell^*}}{1 - \kappa_{\ell^*}} \tag{15}$$

where $\kappa_{\ell^*} = Er((k_{\ell^*}, \ldots, k_r); \mathsf{Ch})$. If $\mathcal{A}_{ur}$ outputs $\mathcal{T}$ such that $T_i \neq \tilde{T}_i$, then $\mathcal{B}$ fails obtaining DL relation if (i) execution of $\mathcal{E}_1'$ returns $\mathsf{tr} = \perp$, or (ii) for the simulated transcript $\rho_I = 0$. The former happens with probability at the most 1-(15) (where $\ell^* = 2$), and the latter happens with probability $(1)/(p)$. The advantage of DL-REL adversary $\mathcal{B}$ is then,

$$\mathbf{Adv}_{\mathsf{GGen}}^{\mathsf{DL\text{-}REL}}(\mathcal{B}) \geq \frac{\epsilon(\mathcal{A}') - (q+1) \cdot \kappa_2}{1 - \kappa_2} - \frac{1}{p}$$

where $\kappa_2 = 1 - \prod_{i=2}^{r} \frac{p-k_i}{p-1}$.

If $\mu \neq \tilde{\mu}$, $\mathcal{B}$ fails in obtaining DL relation only if execution of $\mathcal{E}_1'$ returns $\perp$. The advantage of DL-REL adversary $\mathcal{B}$ is then

$$\mathbf{Adv}_{\mathsf{GGen}}^{\mathsf{DL\text{-}REL}}(\mathcal{B}) \geq \frac{\epsilon(\mathcal{A}') - (q+1)\cdot\kappa_3}{1-\kappa_3}$$

If $\beta_x \neq \tilde{\beta}_x$ or $\hat{t} \neq \tilde{\hat{t}}$, $\mathcal{A}$ always succeeds. Thus,

$$\mathbf{Adv}_{\mathsf{GGen}}^{\mathsf{DL\text{-}REL}}(\mathcal{B}) = \epsilon(\mathcal{A}')$$

If $L_i \neq \tilde{L}_i$ or $R_i \neq \tilde{R}_i$, $\mathcal{B}$ succeeds whenever $\mathcal{E}_1'$ outputs a tree. Thus,

$$\mathbf{Adv}_{\mathsf{GGen}}^{\mathsf{DL\text{-}REL}}(\mathcal{B}) \geq \frac{\epsilon(\mathcal{A}') - (q+1)\cdot\kappa_i}{1-\kappa_i}$$

Finally, if $a \neq \tilde{a}$ or $b \neq \tilde{b}$ then, $\mathcal{B}$ always succeeds.

$$\mathbf{Adv}_{\mathsf{GGen}}^{\mathsf{DL\text{-}REL}}(\mathcal{B}) = \epsilon(\mathcal{A}')$$

Since, all the cases are sequential and $\mathcal{A}_{ur}$ succeeds in forging unless $\mathcal{B}$ breaks discrete-log relation for the very first case that the adversary exploits, it is enough to bound $\mathcal{B}$'s advantage by the minimum rhs value in all the cases. Thus, $\mathcal{B}$'s advantage in breaking DL-REL is at least,

$$\begin{aligned}(1-\kappa_2)\cdot\mathbf{Adv}_{\mathsf{GGen}}^{\mathsf{DL\text{-}REL}}(\mathcal{B}) + \frac{1}{p} &\geq \epsilon(\mathcal{A}') - (q+1)\cdot\kappa_2 \\ &\geq \mathbf{Adv}_{\mathsf{FS\text{-}BP}}^{\mathsf{FS\text{-}WUR}}(\mathcal{A}, \mathcal{S}_{\mathsf{FS\text{-}BP}}) - \frac{1}{p-1} - (q+1)\cdot\kappa_2\end{aligned}$$

Rearranging the terms we obtain the bound in the statement.

Running Time of $\mathcal{B}$. The adversary runs $\mathcal{A}_1$ once to obtain x. Then $\mathcal{B}$ runs $\mathcal{S}_{\mathsf{FS\text{-}BP}}'$ to obtain $\widetilde{\mathcal{T}}$. Finally, $\mathcal{B}$ invokes $\mathcal{E}_1'$ to obtain $\mathcal{T}$ and the tree of transcripts. The run-time of $\mathcal{B}$ is equal to run-time of $\mathcal{A}_1$, plus run-time of $\mathcal{S}_{\mathsf{BP}}'$ (both of which are assumed to be PPT algorithms), plus the running time of $\mathcal{E}_1'$. Clearly, the running time of $\mathcal{E}_1'$ is most dominant. $\mathcal{E}_1'$ terminates after making an expected number of at most $K_2 + q\cdot(K_2-1)$ queries to $\mathcal{A}_2$, where $K_2 = \prod_{i=2}^{r} k_i$, and k_i's are special soundness parameters associated to BP. Thus, $\mathcal{B}$ terminates in expected time $(K_2 + q\cdot(K_2-1))\cdot \mathrm{Time}(\mathcal{A}_2)$. □

Combining the result from Lemma 3.2 we obtain the following corollary.

Corollary 4.1. FS-BP *is FS-SIM-EXT with negligible knowledge error, assuming the hardness of the DL-REL problem. Concretely, let κ be the FS-EXT knowledge error of* FS-BP*, then there exists an adversary $\mathcal{B}$ against DL-REL such that* FS-BP *is FS-SIM-EXT with knowledge error $\hat{\kappa}$ with respect to the NIZK simulator $\mathcal{S}_{\mathsf{FS\text{-}BP}}$, where*

$$\hat{\kappa}(\lambda, q_1, q_2) = q_2\cdot\left((1-\kappa_2)\cdot\mathbf{Adv}_{\mathsf{GGen}}^{\mathsf{DL\text{-}REL}}(\mathcal{B}) + \frac{2}{p-1} + (q_1+1)\cdot\kappa_2\right) + \kappa(\lambda, q_1)$$

and q_1 (resp. q_2) is the number of queries to $\mathcal{S}_1$ (resp. $\mathcal{S}_2'$). The running time of $\mathcal{B}$ is expected polynomial in q_1 and λ.

Acknowledgment

The authors are grateful to Thomas Attema, Matteo Campanelli, Jelle Don, Serge Fehr, Ashrujit Ghoshal, Christian Majenz, and Stefano Tessaro for kindly clarifying the details of their works. We thank anonymous reviewers of EUROCRYPT 2022 for helpful comments about an early version of this work.

The work described in this paper has received funding from: the Concordium Blockhain Research Center, Aarhus University, Denmark; the Carlsberg Foundation under the Semper Ardens Research Project CF18-112 (BCM); the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 803096 (SPEC); Core Research Grant CRG/2020/004488, SERB, Department of Science and Technology; Infosys Young Investigator Award, Infosys Foundation, Bangalore; the Protocol Labs Research Grant Program PL-RGP1-2021-064.

References

AABN02. M. Abdalla, J. H. An, M. Bellare, and C. Namprempre. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In *EUROCRYPT 2002*, vol. 2332 of *LNCS*, pp. 418–433. Springer, Heidelberg, 2002.

ACK21. T. Attema, R. Cramer, and L. Kohl. A compressed Σ-protocol theory for lattices. Cryptology ePrint Archive, Report 2021/307, 2021. https://eprint.iacr.org/2021/307.

AFK21. T. Attema, S. Fehr, and M. Klooß. Fiat-shamir transformation of multi-round interactive proofs. Cryptology ePrint Archive, Report 2021/1377, 2021. https://eprint.iacr.org/2021/1377.

ARS20. B. Abdolmaleki, S. Ramacher, and D. Slamanig. Lift-and-shift: Obtaining simulation extractable subversion and updatable SNARKs generically. In *ACM CCS 2020*, pp. 1987–2005. ACM Press, 2020.

BBB$^+$17. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. Cryptology ePrint Archive, Report 2017/1066, 2017. https://eprint.iacr.org/2017/1066.

BBB$^+$18. 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*, pp. 315–334. IEEE Computer Society Press, 2018.

BCC$^+$16. J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. Cryptology ePrint Archive, Report 2016/263, 2016. https://eprint.iacr.org/2016/263.

BCS16. E. Ben-Sasson, A. Chiesa, and N. Spooner. Interactive oracle proofs. In *TCC 2016-B, Part II*, vol. 9986 of *LNCS*, pp. 31–60. Springer, Heidelberg, 2016.

BDG$^+$13. N. Bitansky, D. Dachman-Soled, S. Garg, A. Jain, Y. T. Kalai, A. López-Alt, and D. Wichs. Why “Fiat-Shamir for proofs” lacks a proof. In *TCC 2013*, vol. 7785 of *LNCS*, pp. 182–201. Springer, Heidelberg, 2013.

BKSV20. 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://ia.cr/2020/811.

BMM$^+$19. B. Bünz, M. Maller, P. Mishra, N. Tyagi, and P. Vesely. Proofs for inner pairing products and applications. Cryptology ePrint Archive, Report 2019/1177, 2019. https://eprint.iacr.org/2019/1177.

CCH$^+$18. R. Canetti, Y. Chen, J. Holmgren, A. Lombardi, G. N. Rothblum, and R. D. Rothblum. Fiat-shamir from simpler assumptions. Cryptology ePrint Archive, Report 2018/1004, 2018. https://eprint.iacr.org/2018/1004.

CCH$^+$19. R. Canetti, Y. Chen, J. Holmgren, A. Lombardi, G. N. Rothblum, R. D. Rothblum, and D. Wichs. Fiat-Shamir: from practice to theory. In *51st ACM STOC*, pp. 1082–1090. ACM Press, 2019.

CDS94. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO’94*, vol. 839 of *LNCS*, pp. 174–187. Springer, Heidelberg, 1994.

DDN91. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography (extended abstract). In *23rd ACM STOC*, pp. 542–552. ACM Press, 1991.

DFM20. J. Don, S. Fehr, and C. Majenz. The measure-and-reprogram technique 2.0: Multi-round fiat-shamir and more. In *CRYPTO 2020, Part III*, vol. 12172 of *LNCS*, pp. 602–631. Springer, Heidelberg, 2020.

DW14. C. Decker and R. Wattenhofer. Bitcoin transaction malleability and MtGox. In *ESORICS 2014, Part II*, vol. 8713 of *LNCS*, pp. 313–326. Springer, Heidelberg, 2014.

FKL18. G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In *CRYPTO 2018, Part II*, vol. 10992 of *LNCS*, pp. 33–62. Springer, Heidelberg, 2018.

FKMV12. S. Faust, M. Kohlweiss, G. A. Marson, and D. Venturi. On the non-malleability of the Fiat-Shamir transform. In *INDOCRYPT 2012*, vol. 7668 of *LNCS*, pp. 60–79. Springer, Heidelberg, 2012.

FS87. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO’86*, vol. 263 of *LNCS*, pp. 186–194. Springer, Heidelberg, 1987.

GK03. S. Goldwasser and Y. T. Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th FOCS*, pp. 102–115. IEEE Computer Society Press, 2003.

GKK$^+$22. C. Ganesh, H. Khoshakhlagh, M. Kohlweiss, A. Nitulescu, and M. Zając. What makes fiat–shamir zksnarks (updatable srs) simulation extractable? Security and Cryptography for Networks, 2022. https://eprint.iacr.org/2021/511.pdf.

GM17. J. Groth and M. Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In *CRYPTO 2017, Part II*, vol. 10402 of *LNCS*, pp. 581–612. Springer, Heidelberg, 2017.

GMR85. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pp. 291–304. ACM Press, 1985.

GOP$^+$21. C. Ganesh, C. Orlandi, M. Pancholi, A. Takahashi, and D. Tschudi. Fiat–shamir bulletproofs are non-malleable (in the algebraic group model). Cryptology ePrint Archive, Report 2021/1393, 2021. https://eprint.iacr.org/2021/1393.

GOP$^+$22. C. Ganesh, C. Orlandi, M. Pancholi, A. Takahashi, and D. Tschudi. Fiat-shamir bulletproofs are non-malleable (in the algebraic group model). In *EUROCRYPT 2022, Part II*, vol. 13276 of *LNCS*, pp. 397–426. Springer, Heidelberg, 2022.

Gro16. J. Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT 2016, Part II*, vol. 9666 of *LNCS*, pp. 305–326. Springer, Heidelberg, 2016.

GT21. A. Ghoshal and S. Tessaro. Tight state-restoration soundness in the algebraic group model. In *CRYPTO 2021, Part III*, vol. 12827 of *LNCS*, pp. 64–93, Virtual Event, 2021. Springer, Heidelberg.

Hol19. J. Holmgren. On round-by-round soundness and state restoration attacks. Cryptology ePrint Archive, Report 2019/1261, 2019. https://eprint.iacr.org/2019/1261.

JT20. J. Jaeger and S. Tessaro. Expected-time cryptography: Generic techniques and applications to concrete soundness. In *TCC 2020, Part III*, vol. 12552 of *LNCS*, pp. 414–443. Springer, Heidelberg, 2020.

KMP16. E. Kiltz, D. Masny, and J. Pan. Optimal security proofs for signatures from identification schemes. In *CRYPTO 2016, Part II*, vol. 9815 of *LNCS*, pp. 33–61. Springer, Heidelberg, 2016.

OO98. K. Ohta and T. Okamoto. On concrete security treatment of signatures derived from identification. In *CRYPTO'98*, vol. 1462 of *LNCS*, pp. 354–369. Springer, Heidelberg, 1998.

PS00. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

Sah99. A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pp. 543–553. IEEE Computer Society Press, 1999.

Wik21. D. Wikström. Special soundness in the random oracle model. Cryptology ePrint Archive, Report 2021/1265, 2021. https://eprint.iacr.org/2021/1265.

A Inner Product Argument

In this section we recall the inner product argument protocol from BP [BBB+17].

Protocol 2: InPRd

$$\mathcal{R}_{\mathsf{InPrd}} = \left\{((n, \mathbf{g}, \mathbf{h}, u), (P), (\mathbf{a}, \mathbf{b})) \mid P = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}u^{c} \wedge c = \langle \mathbf{a}, \mathbf{b} \rangle\right\}$$

1. $\mathcal{P}$ and $\mathcal{V}$ initialize $\mathbf{g}^{(0)} = \mathbf{g}$, $\mathbf{h}^{(0)} = \mathbf{h}$, $n_0 = n$, $P_0 = P$. Additionally, $\mathcal{P}$ initializes $\mathbf{a}^{(0)} = \mathbf{a}$ and $\mathbf{b}^{(0)} = \mathbf{b}$.
2. For $i = \{1, \cdots, \log(n)\}$:
   (a) $\mathcal{P}, \mathcal{V}$ compute:
   $$n_i = n_{i-1}/2$$
   (b) $\mathcal{P}$ computes:
   $$\begin{aligned} c_L &= \left\langle \mathbf{a}^{(i-1)}_{[:n_i]}, \mathbf{b}^{(i-1)}{}_{[n_i:]} \right\rangle, c_R = \left\langle \mathbf{a}^{(i-1)}{}_{[n_i:]}, \mathbf{b}^{(i-1)}_{[:n_i]} \right\rangle \\ L_i &= \left(\mathbf{g}^{(i-1)}_{[n_i:]}\right)^{\mathbf{a}^{(i-1)}[:n_i]} \left(\mathbf{h}^{(i-1)}_{[:n_i]}\right)^{\mathbf{b}^{(i-1)}[n_i:]} u^{c_L} \\ R_i &= \left(\mathbf{g}^{(i-1)}_{[:n_i]}\right)^{\mathbf{a}^{(i-1)}[n_i:]} \left(\mathbf{h}^{(i-1)}_{[n_i:]}\right)^{\mathbf{b}^{(i-1)}[:n_i]} u^{c_R} \end{aligned}$$
   (c) $\mathcal{P}$ sends: L_i, R_i to $\mathcal{V}$.
   (d) $\mathcal{V}$ sends challenge $x_i \xleftarrow{\$} \mathbb{Z}_p^*$.
   (e) $\mathcal{P}, \mathcal{V}$ compute:
   $$\begin{aligned} \mathbf{g}^{(i)} &= \left(\mathbf{g}^{(i-1)}_{[:n_i]}\right)^{x_i^{-1}} \circ \left(\mathbf{g}^{(i-1)}_{[n_i:]}\right)^{x_i} \\ \mathbf{h}^{(i)} &= \left(\mathbf{h}^{(i-1)}_{[:n_i]}\right)^{x_i} \circ \left(\mathbf{h}^{(i-1)}_{[n_i:]}\right)^{x_i^{-1}} \\ P_i &= L_i^{x_i^2} P_{i-1} R_i^{x_i^{-2}} \end{aligned}$$
   (f) $\mathcal{P}$ computes:
   $$\begin{aligned} \mathbf{a}^{(i)} &= \mathbf{a}^{(i-1)}[:n_i]x_i + \mathbf{a}^{(i-1)}[n_i:]x_i^{-1} \\ \mathbf{b}^{(i)} &= \mathbf{b}^{(i-1)}[:n_i]x_i^{-1} + \mathbf{b}^{(i-1)}[n_i:]x_i \end{aligned}$$
3. Let
   $$\begin{aligned} g \leftarrow \mathbf{g}^{(\log(n))}, &\quad h \leftarrow \mathbf{h}^{(\log(n))} \\ a \leftarrow \mathbf{a}^{(\log(n))}, &\quad b \leftarrow \mathbf{b}^{(\log(n))} \end{aligned}$$
   $\mathcal{P}$ sends: a, b to $\mathcal{V}$.
4. $\mathcal{V}$ checks:
   $$P_{\log(n)} \stackrel{?}{=} g^a h^b u^{ab}$$

B Non-Malleability of Bulletproofs – Range Proofs

In this section we recall the range proof protocol from BP [BBB+17] and prove that they satisfy the SR-WUR notion. The arguments follow closely the ones for the proofs for arithemtic circuits.

B.1 Range Proof Protocol

Protocol 3: RngPf

$$\mathcal{R}_{\mathsf{RngPf}} = \{((n, \mathbf{g}, \mathbf{h}, g, h, u), (V), (v, \gamma)) \mid V = g^v h^\gamma \wedge v \in [0, 2^n - 1]\}$$

1. $\mathcal{P}$ computes $\mathbf{a}_L \in \{0,1\}^n$ such that $\langle \mathbf{a}_L, \mathbf{2}^n \rangle = v$, and $\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n$.
2. $\mathcal{P}$ samples $\alpha, \rho \xleftarrow{\$} \mathbb{Z}_p, \mathbf{s}_L, \mathbf{s}_R \xleftarrow{\$} \mathbb{Z}_p^n$, and sends to $\mathcal{V}$:
$$A = h^\alpha \mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R}, \quad S = h^\rho \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R}$$
3. $\mathcal{V}$ sends challenges $y, z \xleftarrow{\$} \mathbb{Z}_p^*$.
4. $\mathcal{P}$ computes:
$$\begin{aligned} l(X) &= (\mathbf{a}_L - z \cdot \mathbf{1}^n) + \mathbf{s}_L \cdot X \\ r(X) &= \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n + \mathbf{s}_R \cdot X) + z^2 \cdot \mathbf{2}^n \\ t(X) &= \langle l(X), r(X) \rangle = t_0 + t_1 \cdot X + t_2 \cdot X \end{aligned}$$
5. $\mathcal{P}$ samples $\beta_1, \beta_2 \xleftarrow{\$} \mathbb{Z}_p$, and sends to $\mathcal{V}$:
$$T_i = g^{t_i} h^{\beta_i}, i \in \{1, 2\}$$
6. $\mathcal{V}$ sends challenge $x \xleftarrow{\$} \mathbb{Z}_p^*$.
7. $\mathcal{P}$ computes:
$$\begin{aligned} &\mathbf{l} = l(x), \quad \mathbf{r} = r(x), \quad \hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle \\ &\beta_x = \beta_2 \cdot x^2 + \beta_1 \cdot x + z^2 \gamma, \quad \mu = \alpha + \rho \cdot x \end{aligned}$$
8. $\mathcal{P}$ sends to $\mathcal{V}$:
$$\beta_x, \mu, \hat{t}$$
9. $\mathcal{V}$ sends challenge $w \xleftarrow{\$} \mathbb{Z}_p^*$.
10. $\mathcal{P}$ and $\mathcal{V}$ compute:
$$\begin{aligned} &\mathbf{h}' = \mathbf{h}^{\mathbf{y}^{-n}}, \quad u' = u^w \\ &P = A \cdot S^x \cdot \mathbf{g}^{-z \cdot \mathbf{1}^n} \cdot \mathbf{h}'^{z \cdot \mathbf{y}^n + z^2 \cdot \mathbf{2}^n}, P' = h^{-\mu} P (u')^{\hat{t}} \end{aligned}$$
11. $\mathcal{P}, \mathcal{V}$ execute inner product argument protocol as $\langle \mathsf{InPrd}.\mathcal{P}((\mathbf{g}, \mathbf{h}', u', P'), (\mathbf{l}, \mathbf{r})), \mathsf{InPrd}.\mathcal{V}(\mathbf{g}, \mathbf{h}', u', P') \rangle$.
12. $\mathcal{V}$ computes:
$$\begin{aligned} \delta(y, z) &= (z - z^2) \cdot \langle \mathbf{1}^n, \mathbf{y}^n \rangle + z^2 \cdot \mathbf{2}^n \\ R &= V^{z^2} g^{\delta(y,z)} T_1^x T_2^{x^2} \\ \mathsf{InPrd}.\mathcal{V}(\mathbf{g}, \mathbf{h}', u', P') &\to b \end{aligned}$$

$\mathcal{V}$ returns 1 if $b = 1$ and $g^{\hat{t}} h^{\beta_x} = R$.

B.2 Alternative Simulation Strategy

In this section we present an alternate simulator $\mathcal{S}'_{\mathsf{RngPf}}$ for RngPf. The simulator works similarly to [BBB+18] but now we explicitly mention how to generate elements A and T_2. This helps in simplifying the proof of SR-WUR for RngPf.

Protocol 4: $\mathcal{S}'_{\mathsf{RngPf}}$

The algebraic simulator $\mathcal{S}_{\mathsf{BP}}$ is given as input:

$$\mathsf{pp} = (n, Q, g, h, u, \mathbf{g}, \mathbf{h}), \quad \mathsf{x} = (V)$$

The transcript is simulated as follows:

1. $x, y, w, z \xleftarrow{\$} \mathbb{Z}_p^*$;
2. $\beta_x, \mu \xleftarrow{\$} \mathbb{Z}_p$;
3. $\mathbf{l}, \mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^n$;
4. $\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle$;
5. $\rho_A, t_2, \beta_2 \xleftarrow{\$} \mathbb{Z}_p$;[a]
6. $A = g^{\rho_A}, T_2 = g^{t_2} h^{\beta_2}$
7. $\mathbf{h}' = \mathbf{h}^{\mathbf{y}^{-n}}; u' = u^w$;
8. $S = \left(A \cdot \mathbf{g}^{-z \cdot \mathbf{1}^n - \mathbf{l}} \cdot (\mathbf{h}')^{z \cdot \mathbf{y}^n + z^2 \cdot \mathbf{2}^n - \mathbf{r}} \cdot h^{-\mu}\right)^{-x^{-1}}$
9. $T_1 = \left(h^{-\beta_x} \cdot g^{(\delta(y,z) - \hat{t})} \cdot V^{z^2} \cdot T_2^{x^2}\right)^{-x^{-1}}$
10. $\mathcal{T} = (S, A; y, z; T_1, T_2; x; \hat{t}, \beta_x, \mu; w; \mathbf{l}, \mathbf{r})$;
11. Output $\mathcal{T}$.

[a] Steps 5,6 are the difference with the original simulator.

Claim 3. *The protocol* RngPf *is perfect HVZK (Definition 2.4) with the simulator* $\mathcal{S}'_{\mathsf{RngPf}}$ *(Protocol 4).*

Proof. Observe that elements A, and T_2 are distributed exactly as in the HVZK simulator for Range proofs in [BBB+18]. Thus, this claim follows from the HVZK proof in [BBB+18].

B.3 Weak Unique Response of FS-RngPf

In this section we show that RngPf is FS-WUR with respect to a NIZK simulator $\mathcal{S}_{\mathsf{FS\text{-}RngPf}}$. Here, $\mathcal{S}_{\mathsf{FS\text{-}RngPf}}$ is the canonical NIZK simulator with respect to $\mathcal{S}_{\mathsf{FS\text{-}RngPf}}$. Let $\mathcal{S}'_{\mathsf{FS\text{-}RngPf}}$ be the canonical NIZK simulator with respect to the simulator $\mathcal{S}'_{\mathsf{FS\text{-}RngPf}}$ explained in the previous subsection.

Claim 4. *Protocol* $\mathsf{FS\text{-}RngPf}$ *satisfies weak unique response (FS-WUR) with respect to a canonical NIZK simulator* $\mathcal{S}_{\mathsf{FS\text{-}RngPf}}$ *in the programmable ROM, under the assumption that solving the discrete-log relation is hard. That is, for every PPT adversary* $\mathcal{A}_{ur}$ *against FS-WUR of* $\mathsf{FS\text{-}RngPf}$ *that makes q random oracle queries, there exists an adversary* $\mathcal{B}$ *against DL-REL that runs in expected polynomial time in q and* λ*, such that,*

$$\mathbf{Adv}^{\mathit{FS\text{-}WUR}}_{\mathsf{FS\text{-}RngPf}}(\mathcal{A}_{ur}, \mathcal{S}_{\mathsf{FS\text{-}RngPf}}) \leq (1 - \kappa_2) \cdot \mathbf{Adv}^{\mathit{DL\text{-}REL}}_{\mathsf{GGen}}(\mathcal{B}) + \frac{2}{p-1} + (q+1) \cdot \kappa_2$$

where $\kappa_2 = 1 - \prod_{i=2}^{r} \frac{p-k_i}{p-1}$ *and* $(k_2, k_3, k_4, \ldots, k_r) = (3, 2, 8, \ldots, 8)$.

Proof. Given an adversary against FS-WUR-game $\mathcal{A}_{ur} = (\mathcal{A}_1, \mathcal{A}_2)$ for protocol $\mathsf{FS\text{-}RngPf}$, we construct an adversary, $\mathcal{B}$, who breaks the discrete-log relation. As in the analysis for $\mathsf{FS\text{-}BP}$, we will first switch from any NIZK simulator $\mathcal{S}_{\mathsf{FS\text{-}RngPf}}$ to the specific one $\mathcal{S}'_{\mathsf{FS\text{-}RngPf}}$. For this we use the same set of hybrids

$\mathsf{G}_0, \mathsf{G}_1$ and a wrapper $\mathcal{A}'$ as defined in the analysis for FS-BP. The only change is that instead of using $\mathcal{S}'_{\text{FS-BP}}$ in $\mathsf{G}_0, \mathsf{G}_1$, here we run the simulator $\mathcal{S}'_{\text{FS-RngPf}}$.

Now, if $\mathcal{A}_{ur}$ produces a winning transcript in G_1 then $\mathcal{B}$ breaks DL-REL as follows. Recall, RngPf is a $(n, 2, 3, 2, 8, \ldots, 8)$-special sound protocol. Thus, by Theorem 2.1 there a series of sub-extractors $(\mathcal{E}'_1, \ldots, \mathcal{E}'_r)$ (same as defined in Fig. 6) where each sub-extractor $(\mathcal{E}'_\ell)$ halts in expected polynomial time and outputs a $(1, \ldots, k_\ell, \ldots, k_r)$-tree of transcripts with probability at least $\frac{\epsilon(\mathcal{A}')-(q+1)\cdot\kappa_\ell}{1-\kappa_\ell}$ (see Lemma 2.1).

Let $\mathcal{T}$ and $\widetilde{\mathcal{T}}$ differ for the first time at the i-th prover message. For the common prefix, $\mathcal{B}$ assumes the same representation as output by $\mathcal{S}'_{\text{FS-RngPf}}$. To obtain the group representation for the rest of the transcript, $\mathcal{B}$ attempts to build a $(1, \ldots, k_{i^*+1}, \ldots, k_r)$-tree by running $\mathcal{E}'_1$. $\mathcal{E}'_1$ either outputs abort or outputs the required tree called tr. Given tr, $\mathcal{B}$ derives representation for $\mathcal{T}$ for all the group elements after and including round $i^* + 1$, using which it breaks discrete-log.

$\mathcal{S}'_{\text{FS-RngPf}}$ outputs transcript of the form:

$$\widetilde{\mathcal{T}} = (\tilde{A}, \tilde{S}; \tilde{y}, \tilde{z}; \tilde{T}_1, \tilde{T}_2; \tilde{x}, \tilde{\beta}_x, \tilde{\mu}, \tilde{\hat{t}}, \tilde{w}, \tilde{L}_1, \tilde{R}_1, \tilde{x}_1, \ldots, \tilde{L}_m, \tilde{R}_m, \tilde{x}_m, \tilde{a}, \tilde{b})$$

and $\mathcal{A}_{ur}$ outputs transcripts of the form:

$$\mathcal{T} = (\tilde{A}, \tilde{S}; \tilde{y}, \tilde{z}; T_1, T_2; x, \beta_x, \mu, \hat{t}, w, L_1, R_1, x_1, \ldots, L_m, R_m, x_m, a, b).$$

Note that at least the first two round messages are the same in both $\mathcal{T}$ and $\widetilde{\mathcal{T}}$. In the following, we do a case by case analysis of $\mathcal{T}$ and $\widetilde{\mathcal{T}}$.

If $\tilde{T}_i \neq T_i$ for some $i \in \{1, 2\}$. Since transcript $\mathcal{T}$ is accepting, the InPrd sub-protocol returns $b = 1$.

$$(\mathbf{g}^{(m)})^a (\mathbf{h}^{(m)})^b (u')^{ab} = \left(\prod_{i=1}^m L_i^{x_i^2}\right) \cdot P' \cdot \left(\prod_{i=1}^m R_i^{x_i^{-2}}\right).$$

$$= \left(\prod_{i=1}^m L_i^{x_i^2}\right) h^{-\mu} \cdot A \cdot S^x \cdot \mathbf{g}^{-z\cdot\mathbf{1}^n} \cdot \mathbf{h}'^{z\cdot\mathbf{y}^n + z^2\cdot\mathbf{2}^n} \cdot (u')^{\hat{t}} \left(\prod_{i=1}^m R_i^{x_i^{-2}}\right)$$

Let us focus on the exponent of generator g. $\mathcal{B}$ derives group representation for L_i, R_i values just as in the proof for Claim 2, and similar to the previous case, the g-component for L_i (resp. R_i) is 0 here as well. The only non-zero g-component is ρ_A from A, and $\rho \cdot (-\tilde{x}^{-1})$ from S.

$$-(\rho_A \cdot \tilde{x}^{-1}) \cdot x + \rho_A \tag{16}$$

If (16) is non-zero then we find a non-trivial discrete-log relation. Otherwise, it must be a zero polynomial. Now consider a subtree rooted at level 2, i.e., it has the same messages T_i's as in $\mathcal{T}$ and then forks for a different challenge $\hat{x}$. Consider a transcript from this subtree. Since it is an accepting transcript as well, we can derive group representation for $\hat{L}_i, \hat{R}_i$ for it. Once again, the coefficient for g and h will come out to be zero. Equating exponents of g, for challenge $\hat{x}$, we get,

$$-(\rho_A \cdot \tilde{\hat{x}}^{-1}) \cdot \hat{x} + \rho_A \tag{17}$$

If either (16) or (17) is non-zero then we find a non-trivial DL solution. Moreover, both cannot be 0 at the same time as this would imply that polynomial $-\rho_A \cdot X^{-1} + \rho_A$ is a zero polynomial, and so $\rho_A = 0$. However, $\rho_A = 0$ happens with probability $1/(p)$ because it is chosen uniformly by the simulator.

If $\beta_x \neq \tilde{\beta}_x$ or $\hat{t} \neq \tilde{\hat{t}}$. This case is the same as the analogous case in Claim 2.

If $\mu \neq \tilde{\mu}$. Similar to (12) in Claim 2, dividing the InPrd verification equation for $\mathcal{T}$ and $\widetilde{\mathcal{T}}$, we get:

$$(\mathbf{g}^{(m)})^a (\mathbf{h}^{(m)})^b (u')^{ab} \cdot (\tilde{\mathbf{g}}^{(m)})^{-\tilde{a}} (\tilde{\mathbf{h}}^{(m)})^{-\tilde{b}} (\tilde{u}')^{-\tilde{a}\tilde{b}}$$
$$= \left(\prod_{i=1}^m L_i^{x_i^2}\right) \left(\prod_{i=1}^m \tilde{L}_i^{-\tilde{x}_i^2}\right) \left(\prod_{i=1}^m R_i^{x_i^{-2}}\right) \left(\prod_{i=1}^m \tilde{R}_i^{-\tilde{x}_i^{-2}}\right) \cdot h^{-(\mu-\tilde{\mu})} \cdot (u')^{\hat{t}} \cdot (\tilde{u}')^{-\hat{t}}.$$

The rest of the analysis is same as in Claim 2.

If $L_i \neq \tilde{L}_i$ or $R_i \neq \tilde{R}_i$. This case is the same as the analogous case in Claim 2.

If $a \neq \tilde{a}$ or $b \neq \tilde{b}$. This case is the same as the analogous case in Claim 2.

Derivation of the concrete advantage as well as the expected running time of $\mathcal{B}$ is analogous to Claim 2. Combining the result from Lemma 3.2 we obtain the simulation-extractability of FS-RngPf analogous to Corollary 4.1.