Tight State-Restoration Soundness in the Algebraic Group Model

Ashrujit Ghoshal and Stefano Tessaro

Paul G. Allen School of Computer Science & Engineering
University of Washington, Seattle, USA
{ashrujit,tessaro}@cs.washington.edu

Abstract. Most efficient zero-knowledge arguments lack a concrete security analysis, making parameter choices and efficiency comparisons challenging. This is even more true for non-interactive versions of these systems obtained via the Fiat-Shamir transform, for which the security guarantees generically derived from the interactive protocol are often too weak, even when assuming a random oracle.
This paper initiates the study of *state-restoration soundness* in the algebraic group model (AGM) of Fuchsbauer, Kiltz, and Loss (CRYPTO '18). This is a stronger notion of soundness for an interactive proof or argument which allows the prover to rewind the verifier, and which is tightly connected with the concrete soundness of the non-interactive argument obtained via the Fiat-Shamir transform.
We propose a general methodology to prove tight bounds on state-restoration soundness, and apply it to variants of Bulletproofs (Bootle et al, S&P '18) and Sonic (Maller et al., CCS '19). To the best of our knowledge, our analysis of Bulletproofs gives the *first* non-trivial concrete security analysis for a non-constant round argument combined with the Fiat-Shamir transform.

Keywords. Zero-knowledge proof systems, concrete security, Fiat-Shamir transform, Algebraic Group Model, state-restoration soundness.

1 Introduction

The last decade has seen zero-knowledge proof systems [GMR85] gain enormous popularity in the design of efficient privacy-preserving systems. Their concrete efficiency is directly affected by the choice of a security parameter, yet *concrete security* analyses are rare and, as we explain below, hit upon technical barriers, even in ideal models (such as the random-oracle [BR93] or the generic-group models [Sho97,Mau05]). This has led to parameter choices not backed by proofs, and to efficiency comparisons across protocols with possibly incomparable levels of security. This paper addresses the question of narrowing this gap for protocols whose security can be analyzed in the Algebraic Group Model [FKL18].

A CONCRETE EXAMPLE. It is convenient to start with an example to illustrate the challenges encountered in proving concrete security of proof systems. We focus on Bulletproofs [BBB+18], which are argument systems with applications across the cryptocurrencies[1] and in verifiably deterministic signatures [NRSW20], which in turn optimize prior work [BCC+16]. The soundness[2] analysis (of their interactive version) is asymptotic, based on the hardness of the *discrete logarithm problem* (DLP). Even when instantiated from 256-bit elliptic curves, due to the absence of a tight, concrete, reduction, we have no formal guarantee on concrete security. Indeed, recent work [JT20] gives concrete soundness bounds in the generic-group model with somewhat unfavorable dependence on the size of the statement being proved, and no better analysis is known.

Even more importantly, existing bounds are for the *interactive* version of the protocol, but Bulletproofs are meant to be used *non-interactively* via the Fiat-Shamir (FS) transform [FS87]. However, the (folklore) analysis of the FS transform gives no useful guarantees: [3] Namely, for a soundness bound ε on the *interactive* ZK proof system, the resulting NIZK has soundness $q^r\varepsilon$, where q is the number of random-oracle queries, and r is the number of challenges sent by the verifier. For Bulletproofs, we have $\varepsilon \geqslant 2^{-256}$ (this is the probability of merely *guessing* the discrete log), and if (say) $r = \Theta(\log(n)) \geqslant 16$, we only get security for (*at best*) $q \leqslant 2^{16}$ queries, which is clearly insufficient.

OVERVIEW OF THIS PAPER. This paper studies the concrete security of succinct proof systems in the *algebraic group model* (AGM) [FKL18], with the goal of developing (near-)exact security bounds. The AGM considers in particular *algebraic* provers that provide representations of group elements to the reduction (or to the extractor), and has been successful to study security in a variety of contexts. More specifically, this work is the first to look at *multi*-round *public-coin* protocols *and* their non-interactive version obtained via the Fiat-Shamir transform. For the latter, we aim for bounds with *linear* degradation in the number of random oracle queries q even for a large number of rounds r, as opposed to the q^r degradation obtained from naïve analyses. Prior work [FKL18] has focused on the simpler case of linear-PCP based SNARKs [Gro16], which are built from two-move interactive proofs and without the FS transform.

The soundness of non-interactive systems resulting from the FS transform is tightly related to the *state-restoration soundness* [BCS16,Hol19] of the underlying interactive protocol, where the cheating prover can *rewind* the verifier as it pleases, until it manages to complete a full accepting interaction with the verifier. No non-trivial bounds on state-restoration soundness are currently known on any non-constant round *argument*.

[1] In particular, Bulletproofs have been deployed in Monero [Mon].

[2] In this introduction, security is with respect to soundness – usually the analysis of zero-knowledge security is much more straightforward.

[3] We are actually *not* aware of any pointer to a write up of this folklore analysis, and we give it for completeness in the paper below

We propose a general framework to quantitatively study state-restoration version of *witness-extended emulation* (wee) [Lin01,GI08] (which implies both state-restoration soundness and a proof-of-knowledge property) in the AGM. We then and apply it to three case studies, which include two variants of Bulletproofs, as well as Sonic [MBKM19]. These protocols have previously been analyzed only with respect to plain soundness in the interactive setting. The analysis of Bulletproofs relies in particular on the Forking Lemma of Bootle *et al.* [BCC+16], which was only very recently made concrete [JT20]. We believe that our framework can be applied to a number of other protocols, such as Hyrax [WTs+18], Dory [Lee20] or pairing-based instantiations of IOPs [BFS20,CHM+20], and leave their analysis for future work.

Remark 1. We stress that our approach differs formally from prior and concurrent works (e.g., [MBKM19,CHM+20]) which use the AGM to give a heuristic validation of the security of a *component* of a protocol, which is then however assumed to satisfy extractability properties compatible with a standard-model proof (i.e., an AGM extractor is used as a standard-model extractor.) Here, we aim for full analyses in the AGM, and as we point out in our technical overview below, these approaches actually do not give a full-fledged proof in the AGM (beyond not giving a proof in the standard model either).

BULLETPROOFS. We apply our framework to two instantiations of Bulletproofs – the first is for *range proofs*, and the other is for general satisfiability of arithmetic circuits. For example, in the former, a prover shows in $O(\log n)$ rounds that for a given Pedersen commitment $C = g^v h^r$ in a cyclic group $\mathbb{G}$ of prime order p we have $v \in [0, 2^n)$. (Here, clearly, $2^n \leqslant p$.)

For the final non-interactive protocol obtained via the FS transform, our result implies that an (algebraic) t-time prover making q random-oracle queries can break security as a Proof of Knowledge (when properly formalized) with advantage roughly

$$\varepsilon(t, q) \leqslant O(qn/p) + \mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(t) \ , \tag{1}$$

where $\mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(t)$ is the advantage of breaking the DLP within time t. In the generic group model, this is roughly $O(t^2/p)$, and this bound justifies the instantiation of Bulletproofs from a 256-bit curve. For arithmetic circuit satisfiability, we obtain a similar bound.

TIGHTNESS AND DISCUSSION. Assuming $\mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(t) \sim t^2/p$ (which is true in the generic group model), the above bound implies in particular that for most values of n,[4] the term $O(qn/p)$ is not leading. Still, we show that the dependence on n is necessary – in particular, we show that there exist n, p for which we can construct a cheating prover that can break soundness with probability $\Omega(qn/p)$, meaning that this part of the bound is tight. (Our argument can be extended to all bounds claimed in the paper.) Also, the term $\mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(t)$ is clearly necessary, given that breaking the DLP would directly give us an attack. This makes our bound essentially exact (up to small constants).

AGM AND COMPOSITION. A challenging aspect of our analysis is the difficulty of dealing with composition. The core of the Bulletproofs is indeed its $O(\log(n))$-round *inner-product argument*. In the standard model, and in the interactive case, it is not hard to reduce the security (as a proof of knowledge) of the full-fledged system using Bulletproofs to the analysis of the underlying inner-product argument, but it is not that clear how to do this generically in the AGM. In particular, in the AGM, the adversary provides representations of group elements to the reduction (or the

[4] For the circuit satisfiability version of our result, one should think of $n = 2^{20}$ and $p = 2^{256}$ as representative values.

extractor), and these are as a function of all priorly given group elements. The problem is that when analyzing a protocol in *isolation* (such as the inner-product argument) the bases to which elements are described are not necessarily the same as those that would be available to a cheating algebraic prover against the *full* protocol. This makes it hard to use an extractor for the inner-product argument in isolation as a sub-routine to obtain an extractor for a protocol using it. Also, because we consider state-restoration soundness, a sub-protocol can be initiated by a cheating prover several times, with several choices of these basis elements.

The downside of this is that our analyses are not modular, at least not at a level which considers sub-protocols are isolated building blocks – we give two different analyses for two different instantiations of Bulletproofs, and the shared modularity is at the algebraic level.

We discuss this further at the end of our technical overview below.

SONIC. As a second application, we study Sonic [MBKM19]. This is a constant-round protocol, and in particular with $3M + 2$ challenges for some constant $M \geqslant 1$. In this case, the folklore analysis of the FS transform can be used to obtain a non-trivial bound, incurring a multiplicative loss of q^{3M+2} from the soundness of the interactive version. Here, we want to show that this loss is not necessary and also obtain a bound which degrades linearly in q. Moreover, no concrete bound on the concrete soundness of Sonic was given in the interactive setting.

We ignore the stronger requirement of updatable witness-extended emulation because our pedagogical point here is that our framework can improve soundness even for constant-round protocols.

We also note that Sonic's proof already uses the AGM to justify security of the underlying polynomial commitment scheme, but follows a (heuristic) pattern described above where the resulting extractor is expected to behave as a standard-model one, and is used within a standard-model proof.

ADAPTIVE VS NON-ADAPTIVE SOUNDNESS. It is important to understand that one can consider both *adaptive* and *non-adaptive* provers, where the former also chooses the *input* for which it attempts to provide a proof. Clearly, one expects adaptive provers to be harder to handle, but this is not necessarily true for *algebraic* provers – in particular, *if* the input contains group elements, the extractor can obtain useful information (and, possibly, directly extract) from their group representation. While this does not render the proof trivial at all, it turns out that for non-adaptive security, the proof is *even harder*. In this paper, we deal mostly with adaptive provers, but for the case of range proofs (where the inputs are commitments in a group), we also give a proof for non-adaptive security – the resulting bound is increased to the square root of the adaptive bound, due to our limited use of rewinding.

RELATED WORK: PROOFS VS ARGUMENTS. We clarify that state-restoration soundness has been studied for several forms of interactive *proofs* [BCS16,Hol19,CCH+18,CCH+19], also in its equivalent form of "round-by-round" soundness. Some proof systems satisfy it directly (such as those based on the sumcheck protocol [LFKN90]), whereas any proof with non-trivial (plain) soundness can be amplified into one with sufficient stare-restoration soundness (e.g., with parallel repetition). This is because (similar to our statement about the Fiat-Shamir transform above) one can naïvely infer that a concrete soundness bound ε implies a state-restoration soundness bound $q^r\varepsilon$, where r is the number of challenges, and thus ε needs to be smaller than q^{-r}.

However, we do not know of any non-trivial bounds on state-restoration soundness for multi-round arguments based on computational assumptions (as opposed to, say, arguments in the ROM), and moreover, soundness amplification (e.g., [Hai09,HPWP10,CL10,BHT20]) does not re-

duce soundness beyond the largest negligible function, and this is insufficient to absorb the q^r loss.

BEYOND THE AGM. Our results are inherently based on online extraction, which is only meaningful in ideal models or using knowledge assumptions. One scenario where ideal models are inherently used is in the compilation of IOPs into NIZKs in the ROM via the BCS transform [BCS16] – it is unclear whether our technique can be used to give tight state-restoration soundness bounds for systems such as Aurora [BCR+19] and STARK [BBHR19].

CONCURRENT WORK. In a recently updated version of [BMM+20], Bünz *et. al.* analyse the soundness of the non-interactive inner-product argument of Bulletproofs in the AGM. We provide a brief comparison with their result in Appendix A, but note here that their analysis is asymptotic, and gives weaker concrete security (insufficient for instantiations on 256-bit curves) when made concrete.

1.1 Overview of our Techniques

We give a general framework to derive tight bounds on state-restoration soundness in the AGM. In fact, we will target the stronger notion of *witness-extended emulation* [Lin01,GI08], which we adapt to state-restoration provers. Recall first that the main characteristic of the AGM is that it allows the reduction, or in our case the extractor, to access representations of group elements. A contribution of independent interest is to set up a formal framework to define extraction in the AGM.

PREFACE: ONLINE EXTRACTION IN THE AGM. In the AGM, the reduction (or an extractor) obtains *representations* of each group element in terms of all previously seen group elements. A useful feature of the AGM is that it often (but not always) allows us to achieve *online witness extraction*, as already observed in [FKL18,FPS20]. In other words, by looking at the representation of the group elements provided by the prover *in a single interaction*, the extractor is able to extract a witness, without the need of rewinding.

Online extraction however immediately appears to be very useful to tame the complexity of state-restoration provers. Indeed, one can visualize an interaction of an adversarial state-restoration prover $\mathcal{P}^*$ with the verifier V as defining an *execution tree*. In particular, $\mathcal{P}^*$ wins if it manages to create a path in the execution tree associated with an accepting (simple) transcript

$$\tau = (a_1, c_1, a_2, \ldots, c_r, a_{r+1}) \ ,$$

where $a_1, a_2, \ldots, a_{r+1}$ are $\mathcal{P}^*$'s messages, and $c_1, \ldots, c_r$ are the verifier's challenges. (We focus on public-coin protocols here.) Online extraction from a single transcript τ *directly* implies extraction here, because a witness can directly be extracted *locally* from the path τ (and the corresponding representations of group elements), disregarding what happened in the rest of the execution tree. In particular, the probability that $\mathcal{P}^*$ succeeds equals the probability that a witness is extracted. Without online extraction, we would have to use rewinding – but current techniques [BCC+16,JT20] do not seem to easily extend to state-restoration provers.

However, this only holds for *perfect* online extraction – in general, we may be able to generate transcripts which are accepting, but for which no witness can be extracted. This is typically because of two reasons:

- **Bad Challenges.** A bad choice of challenges may prevent witness extraction.

- **Violating an assumption.** A transcript is accepting, but the resulting interaction corresponds to a violation of some underlying assumption (i.e., one can extract a non-trivial discrete logarithm relation).

Our framework will exactly follow this pattern. For an r-challenge public-coin protocol, we identify bad challenges, i.e., for each $i \in [r]$, input x, and partial transcript $\tau' = (a_1, c_1, \ldots, a_{i-1}, c_{i-1}, a_i)$, we define a set of bad challenges c_i which would make extraction impossible. Crucially, these sets are defined according to a *simple interaction transcript* (i.e., not a state-restoration one) and can be defined according to the representation of group elements in the transcript so far. Then, given a transcript τ with no bad challenges, we show that:

- We can either extract a witness for x from τ (and the representations of the group elements in τ).
- We can use τ (and the representation of the group elements in terms of the public parameters) to break some underlying assumption.

To illustrate this, we give a non-trivial example next, which considers a simplified instance of the inner product argument at the core of Bulletproofs, but which already captures all subtleties of the model.

INNER-PRODUCT ARGUMENT OF BULLETPROOFS. In the inner product argument the prover proves that a group element $P \in \mathbb{G}$ is a well-formed commitment to vectors $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n$ and their inner-product $\langle \mathbf{a}, \mathbf{b} \rangle$.[5] More precisely, the prover wants to prove to the verifier that $P = \mathbf{g}^{\mathbf{a}} \mathbf{h}^{\mathbf{b}} u^{\langle \mathbf{a}, \mathbf{b} \rangle}$ where $\mathbf{g} \in \mathbb{G}^n, \mathbf{h} \in \mathbb{G}^n, u \in \mathbb{G}$ are independent generators of $\mathbb{G}$.

Here, we shall focus on the special case $n = 2$ first, and below discuss challenges in scaling our analysis up to any n. The prover first sends to the verifier group elements L, R where

$$L = g_2^{a_1} h_1^{b_2} u^{a_1 b_2} \ , \ R = g_1^{a_2} h_2^{b_1} u^{a_2 b_1} \ .$$

The verifier samples x uniformly at random from $\mathbb{Z}_p^*$ and sends it to the prover. We then define

$$P' = L^{x^2} P R^{x^{-2}} \ , \ g' = g_1^{x^{-1}} g_2^{x} \ , \ h' = h_1^{x} h_2^{x^{-1}} \ .$$

The prover sends $a' = a_1 x + a_2 x^{-1}$ and $b' = b_1 x^{-1} + b_2 x$ to the verifier, which in turns accepts if and only if

$$P' = (g')^{a'} (h')^{b'} u^{a' b'} \ .$$

EXTRACTION FOR $n = 2$. For this discussion, we focus in particular on the notion of *adaptive* soundness – i.e., the prover provides P along with its representation, i.e, we get $\mathbf{a}' = (p_{g_1}, p_{g_2})$, $\mathbf{b}' = (p_{h_1}, p_{h_2})$ and p_u such that $P = \mathbf{g}^{\mathbf{a}'} \mathbf{h}^{\mathbf{b}'} u^{p_u}$. At first, it looks like we are done – after all, we can just check whether $\langle \mathbf{a}', \mathbf{b}' \rangle = p_u$, and if so, output $(\mathbf{a}', \mathbf{b}')$ as our witness. Unfortunately, things are not *that* simple – we need to ensure that no accepting transcript $\tau = ((L, R), x, (a', b'))$, i.e., such that $P' = (g')^{a'} (h')^{b'} u^{a' b'}$, is ever produced if $\langle \mathbf{a}', \mathbf{b}' \rangle \neq p_u$, for otherwise our naïve extraction would fail.

To this end, we will prove that if the cheating prover can produce an accepting interaction such while $\langle \mathbf{a}', \mathbf{b}' \rangle \neq p_u$, then we can solve the discrete logarithm problem in the group $\mathbb{G}$. We construct

[5] We use boldface to denote vectors. For two vectors $\mathbf{a} = (a_1, \ldots, a_n), \mathbf{g} = (g_1, \ldots, g_n)$, we use $\mathbf{g}^{\mathbf{a}}$ to denote $\prod_{i=1}^{n} g_i^{a_i}$.

an adversary $\mathcal{A}$ that takes as inputs g_1, g_2, h_1, h_2, u and attempts to return a non-trivial discrete logarithm relation between them. (Breaking this is *tightly* equivalent to breaking the discrete logarithm problem.) Concretely, the adversary $\mathcal{A}$ gives g_1, g_2, h_1, h_2, u as input to the cheating prover $\mathcal{P}$, which first returns an adaptively chosen input $P \in \mathbb{G}$, along with is algebraic representation

$$P = g_1^{p_{g_1}} g_2^{p_{g_2}} h_1^{p_{h_1}} h_2^{p_{h_2}} u^{p_u} \ .$$

The adversary then simulates the execution of $\mathcal{P}$ with a honest verifier further, and assumes it generates an accepting transcript $\tau = ((L, R), x, (a', b'))$ – this transcript contains the representations of L, R such that $L = g_1^{l_{g_1}} g_2^{l_{g_2}} h_1^{l_{h_1}} h_2^{l_{h_2}} u^{l_u}$ and $R = g_1^{r_{g_1}} g_2^{r_{g_2}} h_1^{r_{h_1}} h_2^{r_{h_2}} u^{r_u}$ and since it is an accepting transcript we have

$$L^{x^2} P R^{x^{-2}} = g_1^{x^{-1}a'} g_2^{x^1 a'} h_1^{x^1 b'} h_2^{x^{-1}b'} u^{a'b'} \ .$$

We can plug in the representations of L, R into the equality and obtain values $e_{g_1}, e_{g_2}, e_{h_1}, e_{h_2}, e_u$ such that

$$g_1^{e_{g_1}} g_2^{e_{g_2}} h_1^{e_{h_1}} h_2^{e_{h_2}} u^{e_u} = 1 \ . \tag{2}$$

For example $e_{g_1} = x^{-1}a' - l_{g_1}x^2 - r_{g_1}x^{-2} - p_{g_1}$ and $e_u = a'b' - l_u x^2 - r_u x^{-2} - p_u$.

The adversary $\mathcal{A}$ then simply outputs $(e_{g_1}, e_{g_2}, e_{h_1}, e_{h_2}, e_u)$ – it has found a non-trivial discrete logarithm relation if $(e_{g_1}, e_{g_2}, e_{h_1}, e_{h_2}, e_u) \neq (0, 0, 0, 0, 0)$, which we next show happens with very high probability if $p_u \neq p_{g_1}p_{h_1} + p_{g_2}p_{h_2}$.

Suppose $(e_{g_1}, e_{g_2}, e_{h_1}, e_{h_2}, e_u) = (0, 0, 0, 0, 0)$. From $e_{g_1} = 0$, we have that $x^{-1}a' - l_{g_1}x^2 - r_{g_1}x^{-2} - p_{g_1} = 0$. Since $x \neq 0$, we get that $a' = l_{g_1}x^3 + r_{g_1}x^{-1} + p_{g_1}x$. Similarly from $e_{g_2} = 0$, we would get $a' = l_{g_2}x + p_{g_2}x^{-1} + r_{g_2}x^{-3}$. With high probability over the choice of x's, by the Schwartz-Zippel Lemma, we can infer by equating both right-hand sides that

$$a' = xp_{g_1} + x^{-1}p_{g_2} \ .$$

Similarly, from $e_{h_1} = 0$ and $e_{h_2} = 0$, we obtain that

$$b' = x^{-1}p_{h_1} + xp_{h_2}$$

for most x's. Finally, from $e_u = 0$, we similarly learn that

$$a'b' = x^2 l_u + p_u + x^{-2}r_u \ .$$

Hence from the above

$$x^2 l_u + p_u + x^{-2}r_u = p_{g_1}p_{h_1} + p_{g_2}p_{h_2} + p_{g_1}p_{h_2}x^2 + p_{g_2}p_{h_1}x^{-2} \ .$$

Since we have that $p_{g_1}p_{h_1} + p_{g_2}p_{h_2} \neq p_u$, the above equality holds with very small probability over the choice of x's.

Hence we have shown that $(e_{g_1}, e_{g_2}, e_{h_1}, e_{h_2}, e_u) = (0, 0, 0, 0, 0)$ with very small probability. Therefore $\mathcal{A}$ succeeds with high probability.

NON-ADAPTIVE SECURITY. The above proof exploits the fact that the prover *provides* a representation of P – this corresponds to the case of an *adaptive* prover. But there are scenarios where the prover may be non-adaptive and not be able to do that – for example, the input P has been generated by *another* party, and the prover tries to prove knowledge with respect to this P. It turns out that in this case, one needs a different proof. In fact, one *could* give an extraction strategy which does not require knowing an initial representation for P, but it is then hard to give a reduction to the discrete logarithm problem to show correctness.

We stress that non-adaptive provers and adaptive provers are equivalent in many applications – they only differ when the input includes group elements. We give a formalization and a case study (for Bulletproofs range proofs) in Section 7. There, we can actually give a reduction the discrete logarithm problem (to bound the probability of failing to extract), but this requires rewinding *once* – this allows us to prove a bound which is the square root of the bound for adaptive provers.

THE RECURSIVE PROTOCOL FOR $n = 4$. Scaling the protocol to an arbitrary n proceeds via recursion. For concreteness, let us focus on the case $n = 4$. The prover first sends to the verifier group elements L, R where

$$L = g_3^{a_1} g_4^{a_2} h_1^{b_3} h_2^{b_4} u^{a_1 b_3 + a_2 b_4} \ , \ R = g_1^{a_3} g_2^{a_4} h_3^{b_1} h_4^{b_2} u^{a_3 b_1 + a_4 b_2} \ .$$

The verifier samples x uniformly at random from $\mathbb{Z}_p^*$ and sends it to the prover. The prover and the verifier both compute

$$P' = L^{x^2} P R^{x^{-2}} \ , \ g_1' = g_1^{x^{-1}} g_3^{x} \ , \ g_2' = g_2^{x^{-1}} g_4^{x} \ , \ h_1' = h_1^{x} h_3^{x^{-1}} \ , \ h_2' = h_2^{x} h_4^{x^{-1}} \ .$$

The prover also computes $a_1' = a_1 x + a_3 x^{-1}$, $a_2' = a_2 x + a_4 x^{-1}$, $b_1' = b_1 x^{-1} + b_3 x$ and $b_2' = b_2 x^{-1} + b_4 x$. Observe that

$$P' = (g_1')^{a_1'} (g_2')^{a_2'} (h_1')^{b_1'} (h_3')^{b_2'} u^{a_1' b_1' + a_2' b_2'} \ .$$

Now, the prover and the verifier engage, recursively, in the protocol for $n = 2$ with inputs

$$(g_1', g_2'), (h_1', h_2'), u, P', (a_1', a_2'), (b_1', b_2') \ .$$

The difficulty in analyzing this is that we would like our proof strategy to be recursive, i.e., given we analyzed the protocol for n secure, we can now infer that the one for $2n$ also is secure. This will not be so direct, unfortunately. One major technical issue is for example that the recursive call uses different generators than the ones used for the calling protocol – in our case, here, $(g_1', g_2'), (h_1', h_2')$ – however, when looking at the combined protocol in the AGM, all element representations would be with respect to the generators $g_1, \ldots, g_4, h_1, \ldots, h_4$, and this makes it difficult to directly recycle the above analysis.

THE CHALLENGES WITH COMPOSITION. The inability to leverage recursion to simplify the approach from the previous paragraph is not an isolated incident. We note that a non-trivial aspect of our analyses is due to the lack of easy composition properties in the AGM. In particular, we encounter the following problem – if we have a protocol Π' (e.g., the inner-product argument) which is used as a sub-protocol for Π (a Bulletproofs range proof), and we prove extractability for Π', it is not clear we can infer extractability for Π in a modular way by just calling the extractor for Π'. This is because a stand-alone analysis of Π' may assume group elements output by a malicious prover $\mathcal{P}'$ are represented with respect to some set of basis elements – say, the generators $g_1, \ldots, g_n, h_1, \ldots, h_n, u$ in the concrete example of inner-product argument described above.

However, when Π' is used within Π, the generators of the inner-product argument are functions of *different group* elements. When studying a prover $\mathcal{P}$ attacking Π, then, representations of group elements are with respect to this different set of group elements, and this makes it hard to use an extractor for Π' directly, as it assumes different representations.

This is a problem we encounter in our analyses, and which prevents us from abstracting a theorem for the inner-product argument which we could use, in a plug-and-play way, to imply security of higher-level protocols using it. The flip side is that this lack of composability also comes to our advantage – our extractors will in fact not even need to extract anything from the transcript of an accepting execution of the inner-product argument, but only use the fact that it is accepting to infer correctness of the extracted value.

THE ISSUE WITH PRIOR AGM ANALYSES. Composition issues seemingly affect existing analyses of proof systems in the literature (e.g., [MBKM19,CHM+20]), whenever some components are analyzed in the AGM (typically, a polynomial commitment scheme), but the overall proof is expressed in the standard model. As far as we can tell, unlike this work, one cannot directly extract a full AGM analysis from these works – let us elaborate on this.

Obviously, from a purely formal perspective, the standard model and the algebraic group model cannot be quite mixed, as in particular the AGM extractor for the component cannot be used in the standard model – the only formally correct way to interpret the analysis is as *fully* in the AGM, but part of the analysis does not leverage the full power of the model, and is effectively a standard-model reduction. Yet, in order for composition to be meaningful, it is important to verify that the basis elements assumed in the AGM analysis of the components are the same available to a prover attacking the complete protocol. While we cannot claim any issues (in fact, we give an analysis of Sonic in this paper with a concrete bound), it does appear that all existing works do not attempt to provide a formal composition – they use the existence of an AGM extractor as a heuristic validation for the existence of a standard-model extractor, rather than making formally correct use as an AGM extractor within an AGM proof. Making this composition sound is potentially non-trivial. Having said this, for pairing-based polynomial commitment schemes, the basis elements are generally the same, and thus this can likely be made rigorous fairly easily (unlike the case of inner-product arguments).

2 Preliminaries

Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ represent the set of all natural numbers and let $\mathbb{N}^+ = \mathbb{N}\backslash\{0\}$. For $N \in \mathbb{N}^+$, let $[N] = \{1, \ldots, N\}$. We use $\mathsf{Pr}[\mathsf{G}]$ to denote the probability that the game G returns `true`. Let $\mathbb{G}$ be a cyclic group of prime order p with identity 1 and let $\mathbb{G}^* = \mathbb{G}\backslash\{1\}$ be the set of its generators. We use boldface to denote a vector, e.g., $\mathbf{g} \in \mathbb{G}^n$ is a vector of n group elements with its i^{th} element being g_i, i.e., $\mathbf{g} = (g_1, \ldots, g_n)$. For two vectors $\mathbf{a} = (a_1, \ldots, a_n), \mathbf{g} = (g_1, \ldots, g_n)$, we use $\mathbf{g}^{\mathbf{a}}$ to denote $\prod_{i=1}^n g_i^{a_i}$. We use python notation to denote slices of vectors:

$$\mathbf{g}_{[:l]} = (g_1, \ldots, g_l) \in \mathbb{G}^l \ , \ \mathbf{g}_{[l:]} = (g_{l+1}, \ldots, g_n) \in \mathbb{G}^{n-l} \ .$$

For $z \in \mathbb{Z}_p^*$, we use $\mathbf{z}^n$ to denote the vector $(1, z, z^2, \ldots, z^{n-1})$. Similarly, we use $\mathbf{z}^{-n}$ to denote the vector $(1, z^{-1}, z^{-2}, \ldots, z^{-n+1})$. If Z is a variable, $\mathbf{Z}^n$ represents the vector $(1, Z, Z^2, \ldots, Z^{n-1})$. Our vectors are indexed starting from 1, so $\mathbf{z}_{[1:]}^{n+1}$ is the vector $(z, z^2, \ldots, z^n)$. The operator $\circ$ denotes the Hadamard product of two vectors, i.e.,

$$\mathbf{a} = (a_1, \ldots, a_n) \ , \ \mathbf{b} = (b_1, \ldots, b_n) \ , \ \mathbf{a} \circ \mathbf{b} = (a_1 b_1, \ldots, a_n b_n) \ .$$

Game $\mathsf{G}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{A},\lambda)$:	**Game** $\mathsf{G}^{\mathsf{dl\text{-}rel}}_{\mathbb{G},n}(\mathcal{A},\lambda)$:	**Game** $\mathsf{G}^{q\text{-}\mathsf{dl}}_{\mathbb{G}}(\mathcal{A},\lambda)$:
$g \leftarrow\!\!\$\ \mathbb{G}_\lambda{}^*$; $h \leftarrow\!\!\$\ \mathbb{G}_\lambda$	$g_1,\ldots,g_n \leftarrow\!\!\$\ \mathbb{G}_\lambda$	$g \leftarrow\!\!\$\ \mathbb{G}_\lambda{}^*$
$a \leftarrow\!\!\$\ \mathcal{A}_\lambda(g,h)$	$(a_1,\ldots,a_n) \leftarrow\!\!\$\ \mathcal{A}_\lambda(g_1,\ldots,g_n)$	$x \leftarrow\!\!\$\ \mathbb{Z}_{p(\lambda)}$
Return $(g^a = h)$	Return $(\prod_{i=1}^{n} g_i^{a_i} = 1 \wedge (a_1,\ldots,a_n) \neq \mathbf{0}^n)$	$x' \leftarrow\!\!\$\ \mathcal{A}_\lambda(\{g^x\}^q_{x=-q})$
		Return $(x = x')$

Fig. 1. The games used to define the advantage of a non-uniform adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}^+}$ against the discrete logarithm problem, the discrete logarithm relation problem and the q-DLOG problem in a family of cyclic groups $\mathbb{G} = \{\mathbb{G}_\lambda\}_{\lambda\in\mathbb{N}^+}$ with prime order order $p = p(\lambda)$. The set $\mathbb{G}_\lambda{}^*$ is the set of generators of $\mathbb{G}_\lambda$.

We use capitalized boldface letters to denote matrices, e.g., $\mathbf{W} \in \mathbb{Z}_p^{n\times m}$ is a matrix with n rows and m columns.

We denote the inner product of two vectors $\mathbf{a},\mathbf{b} \in \mathbb{Z}_p^n$ using $\langle \mathbf{a},\mathbf{b}\rangle$. We also define vector polynomials, e.g., $f(X) = \sum_{i=0}^{d} \mathbf{f}_i X^i$, where each coefficient $\mathbf{f}_i$ is a vector in $\mathbb{Z}_p^n$.

The function $\mathsf{bit}(k,i,t)$ returns the bit k_i where $(k_1,\ldots,k_t)$ is the t-bit representation of k.

SCHWARTZ-ZIPPEL LEMMA. The polynomial ring in variables $X_1,\ldots,X_n$ over the field $\mathbb{F}$ is denoted by $\mathbb{F}[X_1,\ldots,X_n]$.

Lemma 1 (Schwartz-Zippel Lemma). *Let $\mathbb{F}$ be a finite field and let $f \in \mathbb{F}[X_1,\ldots,X_n]$ be a non-zero n variate polynomial with maximum degree d. Let S be a subset of $\mathbb{F}$.Then $\mathsf{Pr}\left[f(x_1,\ldots,x_n) = 0\right] \leqslant \frac{d}{|S|}$, where the probability is over the choice of $x_1,\ldots,x_n$ according to $x_i \leftarrow\!\!\$\ S$.*

In particular if p is a prime and $f \in \mathbb{Z}_p[X]$ is a polynomial of degree d and x is sampled uniformly at random from $\mathbb{Z}_p^*$, then $\mathsf{Pr}\left[f(x) = 0\right] \leqslant \frac{d}{p-1}$. Further this implies that if $g(X) = f(X)/X^i$ for $i \in \mathbb{N}$ and x is sampled uniformly at random from $\mathbb{Z}_p^*$, then $\mathsf{Pr}\left[g(x) = 0\right] = \mathsf{Pr}\left[f(x) = 0\right] \leqslant \frac{d}{p-1}$.

THE DISCRETE LOGARITHM PROBLEM. The game $\mathsf{G}^{\mathsf{dl}}_{\mathbb{G}}$ in Figure 1 is used for is used for defining the advantage of a non-uniform adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}^+}$ against the discrete logarithm problem in a family of cyclic groups $\mathbb{G} = \{\mathbb{G}_\lambda\}_{\lambda\in\mathbb{N}^+}$ of prime order $p = p(\lambda)$ with identity 1 and set of generators $\mathbb{G}^* = \{\mathbb{G}^*_\lambda\}_{\lambda\in\mathbb{N}^+} = \{\mathbb{G}_\lambda\backslash\{1\}\}_{\lambda\in\mathbb{N}^+}$. We define

$$\mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{A},\lambda) = \mathsf{Pr}\left[\mathsf{G}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{A},\lambda)\right] .$$

THE DISCRETE LOGARITHM RELATION PROBLEM. The game $\mathsf{G}^{\mathsf{dl\text{-}rel}}_{\mathbb{G},n}$ in Figure 1 is used for defining the advantage of a non-uniform adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}^+}$ against the discrete logarithm relation problem in a family of cyclic groups $\mathbb{G} = \{\mathbb{G}_\lambda\}_{\lambda\in\mathbb{N}^+}$. We define $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}^+}$ as

$$\mathsf{Adv}^{\mathsf{dl\text{-}rel}}_{\mathbb{G},n}(\mathcal{A},\lambda) = \mathsf{Pr}\left[\mathsf{G}^{\mathsf{dl\text{-}rel}}_{\mathbb{G},n}(\mathcal{A},\lambda)\right] .$$

The following lemma shows that hardness of the discrete logarithm relation problem in $\mathbb{G}$ is tightly implied by the hardness of discrete logarithm problem in a family of cyclic groups $\mathbb{G} = \{\mathbb{G}_\lambda\}_{\lambda\in\mathbb{N}^+}$.

Lemma 2. *Let $n \in \mathbb{N}^+$. Let $\mathbb{G} = \{\mathbb{G}_\lambda\}_{\lambda\in\mathbb{N}^+}$ be a family of cyclic groups with order $p = p(\lambda)$. For every non-uniform adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}^+}$ there exists a non-uniform adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_{\lambda\in\mathbb{N}^+}$ such that for all $\lambda \in \mathbb{N}^+$ $\mathsf{Adv}^{\mathsf{dl\text{-}rel}}_{\mathbb{G},n}(\mathcal{A},\lambda) \leqslant \mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{B},\lambda) + \frac{1}{p}$. Moreover, $\mathcal{B}$ is nearly as efficient as $\mathcal{A}$.*

We refer the reader to [JT20] for a proof of this lemma.

Game $\mathsf{SRS}^{\mathcal{P}}_{\mathsf{IP}}(\lambda)$:	**Oracle** $\mathbf{O}_{\mathrm{ext}}(\tau = (a_1, c_1, \ldots, a_{i-1}, c_{i-1}), a_i)$:
$\mathsf{win} \leftarrow \mathtt{false}$; $\mathsf{tr} \leftarrow \varepsilon$	If $\tau \in \mathsf{tr}$ then
$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{IP.Setup}(1^\lambda)$	If $i \leqslant r$ then
$(x, \mathsf{st}_{\mathcal{P}}) \leftarrow\!\!\$\ \mathcal{P}_\lambda(\mathsf{pp})$	$c_i \leftarrow\!\!\$\ \mathsf{Ch}_i$; $\mathsf{tr} \leftarrow \mathsf{tr} \parallel (\tau, a_i, c_i)$; Return c_i
Run $\mathcal{P}_\lambda^{\mathbf{O}_{\mathrm{ext}}}(\mathsf{st}_{\mathcal{P}})$	Else if $i = r + 1$ then
Return win	$d \leftarrow \mathsf{IP.V}(\mathsf{pp}, x, (\tau, a_i))$; $\mathsf{tr} \leftarrow \mathsf{tr} \parallel (\tau, a_i)$
	If $d = 1$ then $\mathsf{win} \leftarrow \mathtt{true}$
	Return d
	Return $\perp$

Fig. 2. Definition of state-restoration soundness. The game SRS defines state-restoration soundness for a non-uniform prover $\mathcal{P}$ and a public-coin interactive proof IP. Here, IP has $r = r(\lambda)$ challenges and the i-th challenge is sampled from Ch_i.

THE q-DLOG PROBLEM. The game $\mathsf{G}^{q\text{-dl}}_{\mathbb{G}}$ in Figure 1 is used for defining the advantage of a non-uniform adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}^+}$ against the q-DLOG problem in a family of groups $\mathbb{G} = \{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}^+}$. We define

$$\mathsf{Adv}^{q\text{-dl}}_{\mathbb{G}}(\mathcal{A}, \lambda) = \mathsf{Pr}\left[\mathsf{G}^{q\text{-dl}}_{\mathbb{G}}(\mathcal{A}, \lambda)\right] .$$

We note that there are other problems known as q-DLOG which are not equivalent to the one we use here. We use the version stated above because it was the version used in the analysis of Sonic [MBKM19] which we analyse in this paper.

3 Interactive Proofs and State-restoration Soundness

We introduce our formalism for handling interactive proofs and arguments, which is particularly geared towards understanding their concrete state-restoration soundness.

INTERACTIVE PROOFS. An *interactive proof* [GMR85] IP is a triple of algorithms: (1) the *setup algorithm* IP.Setup which generates the public parameters pp, (2) the *prover* IP.P and (3) the *verifier* IP.V. In particular, the prover and the verifier are interactive machines which define a two-party protocol, where the prover does not produce any output, and the verifier outputs a decision bit $d \in \{0, 1\}$. We let $\langle \mathsf{IP.P}(x), \mathsf{IP.V}(y) \rangle$ denote the algorithm which runs an execution of the prover and the verifier on inputs x and y, respectively, and outputs the verifier's decision bit. We say that IP is *public coin* if all messages sent from IP.V to IP.P are fresh random values from some understood set (which we refer to as *challenges*).

COMPLETENESS. A *relation* R is (without loss of generality) a subset of $\{0,1\}^* \times \{0,1\}^* \times \{0,1\}^*$. We denote a relation R that uses specified public parameters pp, instance x and witness w as $\{(\mathsf{pp}, x, w) : f_R(\mathsf{pp}, x, w)\}$ where $f_R(\mathsf{pp}, x, w)$ is a function that returns $\mathtt{true}$ if $(\mathsf{pp}, x, w) \in R$ and $\mathtt{false}$ otherwise. For every $\lambda \in \mathbb{N}^+$ and every $\mathcal{A}$, define the following experiment:

$$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{IP.Setup}(1^\lambda) , \quad (x, w) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{pp}) , \quad d \leftarrow\!\!\$\ \langle \mathsf{IP.P}(\mathsf{pp}, x, w), \mathsf{IP.V}(\mathsf{pp}, x) \rangle .$$

Then, we say that IP is an interactive proof for the relation R if for all $\mathcal{A}$ and all $\lambda \in \mathbb{N}^+$, in the above experiment the event $(d = 1) \vee ((\mathsf{pp}, x, w) \notin R)$ holds with probability one.

STATE-RESTORATION SOUNDNESS. We target a stronger notion of soundness – *state-restoration soundness* (SRS) [BCS16,Hol19] – which (as we show below) tightly reduces to the soundness of the non-interactive proof obtained via the Fiat-Shamir transform. The SRS security game allows

the cheating prover to *rewind* the verifier as it pleases, and wins if and only if it manages to produce *some* accepting interaction. We only consider an $r(\lambda)$-challenge *public-coin* interactive proof IP, and consider the case where challenges are drawn uniformly from some sets $\mathsf{Ch}_1, \ldots, \mathsf{Ch}_r$. We also assume that the verifier is described by an algorithm which given pp, x, and a *transcript* $\tau = (a_1, c_1, \ldots, a_r, c_r, a_{r+1})$, outputs a decision bit $d \in \{0,1\}$. We overload notation and write $\mathsf{IP.V}(\mathsf{pp}, x, \tau)$ for this output.

Our definition considers a game $\mathsf{SRS}^{\mathcal{P}}_{\mathsf{IP}}(\lambda)$ (which is formalized in Figure 2) that involves a non-uniform cheating prover $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda\in\mathbb{N}}$. (Henceforth, whenever we have any non-uniform adversary $\mathcal{A}$, it is understood $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ – we shall not specify this explicitly). The prover is initially responsible for generating the input x on which it attempts to convince the verifier on *some* execution. Its rewinding access to the verifier is ensured by an oracle $\mathbf{O}_{\mathrm{ext}}$, to which it has access. Roughly speaking, the oracle allows the prover to build an *execution tree*, which is extended with each query to it by the prover. This execution tree can be inferred from tr, which sequentially logs all (valid) queries to $\mathbf{O}_{\mathrm{ext}}$ by the prover. For a partial transcript τ', we write $\tau' \in \mathsf{tr}$ to mean that a partial execution corresponding to τ' can be inferred from tr.

We then associate the probability of winning the game with the srs *advantage metric*,

$$\mathsf{Adv}^{\mathsf{srs}}_{\mathsf{IP}}(\mathcal{P}, \lambda) = \mathsf{Pr}\left[\mathsf{SRS}^{\mathsf{IP}}_{\mathcal{P}}(\lambda)\right] .$$

For notational convenience, we do not restrict the input x not to have a witness. Therefore, if IP is an interactive proof for a relation R, we cannot hope to show that $\mathsf{Adv}^{\mathsf{srs}}_{\mathsf{IP}}(\mathcal{P}, \lambda)$ is small *for all* $\mathcal{P}$. Clearly, if $\mathcal{P}$ outputs (x, a) such that $(\mathsf{pp}, x, a) \in R$, then a is a witness and $\mathcal{P}$ can simply (honestly) convince the verifier. The classical notion of state-restoration soundness is recovered by only considering $\mathcal{P}$'s which output x such that $(\mathsf{pp}, x, w) \notin R$ for any w.

The following lemma shows a (generally loose) connection between (plain) soundness and state restoration soundness.

Lemma 3 (Naïve Reduction). *Let* IP *be a* $r(\lambda)$*-challenge public-coin interactive proof. Then, for every non-uniform prover* $\mathcal{P}$ *invoking* $\mathbf{O}_{\mathrm{ext}}$ *at most* $q = q(\lambda)$ *times, there exists a* linear *prover* $\mathcal{P}'$ *(with complexity similar to that of* $\mathcal{P}$*) such that for all* $\lambda \in \mathbb{N}^+$, $\mathsf{Adv}^{\mathsf{srs}}_{\mathsf{IP}}(\mathcal{P}, \lambda) \leqslant \binom{q(\lambda)}{r(\lambda)+1} \cdot \mathsf{Adv}^{\mathsf{srs}}_{\mathsf{IP}}(\mathcal{P}', \lambda)$.

We omit the (simple) proof – the adversary $\mathcal{P}'$ simply "guesses" the accepting path, which consists of $r+1$ queries.

If IP is publicly verifiable, we can prove the following slightly improved bound.$\mathsf{Adv}^{\mathsf{srs}}_{\mathsf{IP}}(\mathcal{P}, \lambda) \leqslant \binom{q(\lambda)}{r(\lambda)} \cdot \mathsf{Adv}^{\mathsf{srs}}_{\mathsf{IP}}(\mathcal{P}', \lambda)$. In this case the adversary $\mathcal{P}'$ would need to guess only the first r messages and use the public verification procedure to check if any of the q queries is a valid last message.

4 Proofs of Knowledge in the AGM

THE ALGEBRAIC GROUP MODEL. We start here with a brief review of the AGM [FKL18]. For an understood group $\mathbb{G}$ with prime order p, an *algebraic* algorithm $\mathcal{A}_{\mathsf{alg}}$ is an interactive algorithm whose inputs and outputs are made of distinct group elements and strings. Furthermore, each (encoding) of a group element X output by $\mathcal{A}_{\mathsf{alg}}$ is accompanied by a *representation* $(x_{A_1}, x_{A_2}, \ldots, x_{A_k}) \in \mathbb{Z}_p^k$ such that $X = \prod_{i=1}^{k} A_i^{x_{A_i}}$, where $A_1, \ldots, A_k$ are all group elements previously input *and* output by $\mathcal{A}_{\mathsf{alg}}$. Generally, we denote a group element by itself with a capital letter X, and write $[X]$ for a group element X *enhanced* with its representation, e.g.,$[X] = (X, x_{A_1}, x_{A_2}, \ldots, x_{A_k})$. In particular,

Game $\mathsf{WEE\text{-}1}^{\mathcal{P}_{\mathsf{alg}},\mathcal{D}}_{\mathsf{IP}}(\lambda)$:
$\mathsf{tr} \leftarrow \varepsilon$
$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{IP.Setup}(1^\lambda)$
$([x], \mathsf{st}_{\mathcal{P}}) \leftarrow\!\!\$\ \mathcal{P}_{\mathsf{alg},\lambda}(\mathsf{pp})$
Run $\mathcal{P}^{\mathbf{O}^1_{\mathrm{ext}}}_{\mathsf{alg},\lambda}(\mathsf{st}_{\mathcal{P}})$
$b \leftarrow\!\!\$\ \mathcal{D}(\mathsf{tr})$
Return $(b = 1)$

Game $\mathsf{WEE\text{-}0}^{\mathcal{E},\mathcal{P}_{\mathsf{alg}},\mathcal{D}}_{\mathsf{IP},R}(\lambda)$:
$\mathsf{tr} \leftarrow \varepsilon$
$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{IP.Setup}(1^\lambda)$
$([x], \mathsf{st}_{\mathcal{P}}) \leftarrow\!\!\$\ \mathcal{P}_{\mathsf{alg},\lambda}(\mathsf{pp})$
$\mathsf{st}_{\mathcal{E}} \leftarrow (1^\lambda, \mathsf{pp}, [x])$
Run $\mathcal{P}^{\mathbf{O}^0_{\mathrm{ext}}}_{\mathsf{alg},\lambda}(\mathsf{st}_{\mathcal{P}})$
$w \leftarrow\!\!\$\ \mathcal{E}(\mathsf{st}_{\mathcal{E}}, \bot)$
$b \leftarrow\!\!\$\ \mathcal{D}(\mathsf{tr})$
Return $(b = 1) \wedge (\mathsf{Acc}(\mathsf{tr}) \Rightarrow (\mathsf{pp}, x, w) \in R)$

Oracle $\mathbf{O}^1_{\mathrm{ext}}(\tau = (a_1, c_1, \ldots, a_{i-1}, c_{i-1}), a_i)$:
If $\tau \in \mathsf{tr}$ then
  If $i \leqslant r$ then
    $c_i \leftarrow\!\!\$\ \mathsf{Ch}_i$; $\mathsf{tr} \leftarrow \mathsf{tr} \,\|\, (\tau, a_i, c_i)$; return c_i
  Else if $i = r + 1$ then
    $d \leftarrow \mathsf{IP.V}(\mathsf{pp}, x, \tau \,\|\, a_i)$
    If $d = 1$ then return d
Return $\bot$

Oracle $\mathbf{O}^0_{\mathrm{ext}}(\tau = (a_1, c_1, \ldots, a_{i-1}, c_{i-1}), a_i)$:
If $\tau \in \mathsf{tr}$ then
  If $i \leqslant r$ then
  $(\mathsf{resp}, \mathsf{st}_{\mathcal{E}}) \leftarrow\!\!\$\ \mathcal{E}(\mathsf{st}_{\mathcal{E}}, [(\tau, a_i)])$
  $\mathsf{tr} \leftarrow \mathsf{tr} \,\|\, (\tau, a_i, \mathsf{resp})$
  Return resp
  Return $\bot$

Fig. 3. Definition of online srs-wee security in the AGM. The games $\mathsf{WEE}\text{-}1, \mathsf{WEE}\text{-}0$ define online srs-wee security in the AGM for a non-uniform algebraic prover $\mathcal{P}_{\mathsf{alg}}$, a distinguisher $\mathcal{D}$, an extractor $\mathcal{E}$ and a public-coin interactive proof IP. We assume here that IP has $r = r(\lambda)$ challenges and the i-th challenge is sampled from Ch_i.

when we use a group element X output by $\mathcal{A}_{\mathsf{alg}}$, e.g. it is *input* to a reduction or used in a cryptographic game, we write $[X]$ to make explicit that the representation is available, whereas write X only when the representation is omitted. The notation extends to a mix of group elements and strings a – $[a]$ enhances each group elements with its representation.

DEFINING AGM EXTRACTION. We formalize a notion of proof-of-knowledge (PoK) security in the AGM, following the lines of witness-extended emulation [Lin01,GI08], which we extend to provers that can rewind the verifier.

We will be interested in cases where the AGM allows for online extraction, i.e., the additional group representations will allow for extraction without rewinding the prover. (Note that the prover itself can rewind the verifier, which is a little different.) We target an adaptive notion of security, where the input is generated by the adversarial prover *itself*, depending on the public parameters pp, and can contain group elements.

ONLINE SRS-WEE SECURITY. The definition consists of two games – denoted $\mathsf{WEE\text{-}1}^{\mathcal{P}_{\mathsf{alg}},\mathcal{D}}_{\mathsf{IP}}$ and $\mathsf{WEE\text{-}0}^{\mathcal{E},\mathcal{P}_{\mathsf{alg}},\mathcal{D}}_{\mathsf{IP},R}$, and described in Figure 3. The former captures the real game, lets our prover $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda\in\mathbb{N}}$ interact with an oracle $\mathbf{O}^1_{\mathrm{ext}}$ as in the state-restoration soundness game defined above, which additionally stores a transcript tr. The latter is finally given to a *distinguisher* $\mathcal{D}$ which outputs a decision bit. In contrast, the *ideal* game delegates the role of answering $\mathcal{P}$'s oracle queries to a (stateful) extractor $\mathcal{E}$. The extractor, at the end of the execution, also outputs a witness candidate for w. The extractor in particular exploits here the fact that $\mathcal{P}$ is algebraic by learning the representation of every input to the oracle $\mathbf{O}^0_{\mathrm{ext}}$. (This representation can be thought, without loss of generality, as being in terms of all group elements contained in pp.) Here, the final output of the game is not merely $\mathcal{D}$'s decision bit – should the latter output 1, the output of the game is $\mathtt{true}$ only if additionally the extracted witness is correct assuming the interaction with $\mathbf{O}^0_{\mathrm{ext}}$ resulted in an accepting execution – a condition we capture via the predicate $\mathsf{Acc}(\mathsf{tr})$.

For an interactive proof IP and an associated relation R, non-uniform algebraic prover $\mathcal{P}_{\mathsf{alg}}$, a distinguisher $\mathcal{D}$, and an extractor $\mathcal{E}$, we define

$$\mathsf{Adv}^{\mathsf{sr\text{-}wee}}_{\mathsf{IP},R}(\mathcal{P}_{\mathsf{alg}},\mathcal{D},\mathcal{E},\lambda) = \mathsf{Pr}\left[\mathsf{WEE\text{-}1}^{\mathcal{P}_{\mathsf{alg}},\mathcal{D}}_{\mathsf{IP}}(\lambda)\right] - \mathsf{Pr}\left[\mathsf{WEE\text{-}0}^{\mathcal{E},\mathcal{P}_{\mathsf{alg}},\mathcal{D}}_{\mathsf{IP},R}(\lambda)\right] . \tag{3}$$

One can consider also scenarios where the prover may be non-adaptive – for example, the input has been generated by *another* party, and the prover tries to prove knowledge with respect to this input. For this reason, introduce the notion of non-adaptive srs-wee in Section 7.

SOUNDNESS FROM POK. The definition of state-restoration soundness from Section 3 also applies to any algebraic prover. The following theorem relates soundness to the witness-extended emulation – the proof is immediate.

Lemma 4. *Let IP be an interactive proof for a relation R, and let $\mathcal{P}_{\mathsf{alg}}$ an algebraic prover which, on input pp, outputs x such that $(\mathsf{pp},x,w)\notin R$ for all w. Then, for any extractor $\mathcal{E}$, and $\mathcal{D}(\cdot)=\mathsf{Acc}(\cdot)$, we have for all $\lambda\in\mathbb{N}^+$, $\mathsf{Adv}^{\mathsf{srs}}_{\mathsf{IP}}(\mathcal{P}_{\mathsf{alg}},\lambda)\leqslant\mathsf{Adv}^{\mathsf{sr\text{-}wee}}_{\mathsf{IP},R}(\mathcal{P}_{\mathsf{alg}},\mathcal{D},\mathcal{E},\lambda)$.*

4.1 The Basic Framework

We develop a general framework that we will use, via Theorem 1, to derive concrete AGM bounds on srs-wee security. Our goal, in particular, is to give conditions on *single* path executions – i.e., executions not involving any rewinding of the verifier by the prover, which could be seen as root-to-leaf paths in an execution tree generated by the interaction of a state-restoration prover.

TRANSCRIPTS. From now on, let us fix an interactive *public-coin* proof $\mathsf{IP}=(\mathsf{IP.Setup},\mathsf{IP.P},\mathsf{IP.V})$ for a relation R. Assume further this protocol has exactly r rounds of challenges. Then, we represent a (potential) *single-execution* transcript generated by an algebraic prover in different forms, depending on whether we include the representations of group elements or not. Specifically, we let the (plain) transcript be $\tau=(\mathsf{pp},x,a_1,c_1,a_2,c_2,\ldots,a_r,c_r,a_{r+1})$, where pp are the generated parameters, x is the input produced by $\mathcal{P}_{\mathsf{alg}}$, $c_i\in\mathsf{Ch}_i$ for all $i\in\{1,\ldots,r\}$ are the challenges, and $a_1,\ldots,a_{r+1}$ are the prover's messages. The corresponding *extended transcript* with representations is denoted as $[\tau]=(\mathsf{pp},[x],[a_1],c_1,[a_2],c_2,\ldots,[a_r],c_r,[a_{r+1}])$.

In particular, the representation of each group element contained in a_i is with respect to all elements contained in $\mathsf{pp},x,a_1,\ldots,a_{i-1}$. We let $\mathcal{T}^{\mathsf{IP}}$ be the set of all possible extended transcripts $[\tau]$. We also let $\mathcal{T}^{\mathsf{IP}}_{\mathsf{Acc}}\subseteq\mathcal{T}^{\mathsf{IP}}$ be the set of *accepting* transcripts $[\tau]$, i.e., $\mathsf{IP.V}(\tau)=1$.

PATH EXTRACTION. We now would like to define a function e which extracts a witness from any accepting transcript $[\tau]\in\mathcal{T}^{\mathsf{IP}}_{\mathsf{Acc}}$. For a particular function e we now define the set of extended transcripts on which it succeeds in extracting a valid witness, i.e.,

$$\mathcal{T}^{\mathsf{IP},\mathsf{e},R}_{\mathsf{correct}} = \left\{[\tau]=(\mathsf{pp},[x],\ldots)\in\mathcal{T}^{\mathsf{IP}}_{\mathsf{Acc}} \;:\; w\leftarrow\mathsf{e}([\tau]),\ (\mathsf{pp},x,w)\in R\right\} .$$

Therefore, a natural extractor $\mathcal{E}$ just answers challenges honestly, and applies e to a path in the execution tree which defines an accepting transcript, and returns the corresponding witness w. The probability of this extractor failing can be upper bounded naïvely by the probability that the prover generates, in its execution tree, a path corresponding to an extended transcript $[\tau]\in\mathcal{T}^{\mathsf{IP}}_{\mathsf{Acc}}\setminus\mathcal{T}^{\mathsf{IP},\mathsf{e},R}_{\mathsf{correct}}$. This is however not directly helpful, as the main challenge is to actually estimate this probability.

BAD CHALLENGES. In all of our examples, the analysis of the probability of generating a transcript in $\mathcal{T}_{\mathsf{Acc}}^{\mathsf{IP}} \setminus \mathcal{T}_{\mathsf{correct}}^{\mathsf{IP},\mathsf{e},R}$ will generally consist of an *information-theoretic* and a *computational part*.

The information-theoretic part will account to choosing some *bad challenges*. We capture such choices of bad challenges by defining, for any partial extended transcript

$$\left[\tau'\right] = (\mathsf{pp}, [x], [a_1], c_1, \ldots, [a_i]) ,$$

a set $\mathsf{BadCh}(\tau') \subseteq \mathsf{Ch}_i$ of such bad challenges. (Crucially, whether a challenge is bad or not only depends on the extended transcript so far.) We now denote as $\mathcal{T}_{\mathsf{BadCh}}^{\mathsf{IP}}$ the set of all extended transcripts which contain at least one bad challenge. It turns out that the probability of generating such a bad challenge is easily bounded by $q \cdot \varepsilon$ for a prover making q oracle queries, assuming $\left|\mathsf{BadCh}(\tau')\right| / \left|\mathsf{Ch}_i\right| \leqslant \varepsilon$.

The only case that the extractor can now fail is if the execution tree contains an extended transcript $[\tau]$ in the set $\mathcal{T}_{\mathsf{fail}}^{\mathsf{IP},\mathsf{e},R} = \mathcal{T}_{\mathsf{Acc}}^{\mathsf{IP}} \setminus (\mathcal{T}_{\mathsf{correct}}^{\mathsf{IP},\mathsf{e},R} \cup \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{IP}})$. We denote the probability that this happens in $\mathsf{SRS}_{\mathsf{IP}}^{\mathcal{P}_{\mathsf{alg}}}(\lambda)$ as $p_{\mathsf{fail}}(\mathsf{IP}, \mathcal{P}_{\mathsf{alg}}, \mathsf{e}, R, \lambda)$. Generally, in all of our applications, upper bounding this probability for a suitably defined extractor will constitute the computational core of the proof – i.e., we will prove (generally tight) reductions to breaking some underlying assumption.

THE MASTER THEOREM. We are now ready to state our master theorem, which assumes the formal set up.

Theorem 1 (Master Theorem). *Let* IP *be an* $r = r(\lambda)$*-challenge public coin interactive proof for a relation* R*. Assume that* BadCh *and* e *are as given above. Let* τ' *be a partial transcript such that the challenge that comes right after is sampled from* Ch_i*. Assume that for all* $i \in \{1, \ldots, r\}$*, we have*

$$\left|\mathsf{BadCh}(\tau')\right| / \left|\mathsf{Ch}_i\right| \leqslant \varepsilon$$

for some $\varepsilon \in [0, 1]$*. Then, there exists an extractor* $\mathcal{E}$ *that uses* e *such that for any non-uniform algebraic prover* $\mathcal{P}_{\mathsf{alg}}$ *making at most* $q = q(\lambda)$ *queries to its oracle, and any (computationally unbounded) distinguisher* $\mathcal{D}$*, for all* $\lambda \in \mathbb{N}^+$

$$\mathsf{Adv}_{\mathsf{IP},R}^{\mathsf{sr\text{-}wee}}(\mathcal{P}_{\mathsf{alg}}, \mathcal{D}, \mathcal{E}, \lambda) \leqslant q\varepsilon + p_{\mathsf{fail}}(\mathsf{IP}, \mathcal{P}_{\mathsf{alg}}, \mathsf{e}, R, \lambda) .$$

The time complexity of the extractor $\mathcal{E}$ *is* $O(q \cdot t_V + t_{\mathsf{e}})$ *where* t_V *is the time required to run* $\mathsf{IP.V}$ *and* t_{e} *is the time required to run* e*.*

Proof. The extractor $\mathcal{E}$, as stated in Section 4.1, just answers challenges honestly, and applies e to a path in the execution tree which defines an accepting transcript, and returns whatever e returns. The running time of the extractor $\mathcal{E}$ consists of the time required to answers q queries, run $\mathsf{IP.V}$ in at most q paths in the execution tree and the time required to run e. Hence it's time complexity is $O(q \cdot t_V + t_{\mathsf{e}})$.

Since, $\mathcal{E}$ answers challenges honestly, the view of $\mathcal{P}_{\mathsf{alg}}$ is identical in the games $\mathsf{WEE\text{-}1}_{\mathsf{IP}}^{\mathcal{P}_{\mathsf{alg}},\mathcal{D}}$ and $\mathsf{WEE\text{-}0}_{\mathsf{IP},R}^{\mathcal{E},\mathcal{P}_{\mathsf{alg}},\mathcal{D}}$. So, tr will be identical in both games and hence b will be identical in both games. Therefore, the output of $\mathsf{WEE\text{-}0}_{\mathsf{IP},R}^{\mathcal{E},\mathcal{P}_{\mathsf{alg}},\mathcal{D}}$ differs from the output of $\mathsf{WEE\text{-}1}_{\mathsf{IP}}^{\mathcal{P}_{\mathsf{alg}},\mathcal{D}}$ only if $(\mathsf{Acc}(\mathsf{tr}) \Rightarrow (\mathsf{pp}, x, w) \in R) = \mathtt{false}$ i.e., if $\mathsf{Acc}(\mathsf{tr})$ is $\mathtt{true}$ but $(\mathsf{pp}, x, w) \notin R$.

Since $\mathsf{Acc}(\mathsf{tr})$ is $\mathtt{true}$, there is an accepting transcript τ such that $\mathcal{E}$ gives $[\tau]$ as input to e. Now, e outputs w such that $(\mathsf{pp}, x, w) \notin R$ only if $\tau \in \mathcal{T}_{\mathsf{fail}}^{\mathsf{IP},\mathsf{e},R}$ or $\tau \in \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{IP}}$ (these sets are defined in Section 4.1).

Game $\mathsf{FS\text{-}EXT\text{-}1}^{\mathcal{P}_{\mathsf{alg}},\mathcal{E}}_{\mathsf{IP},R}(\lambda)$:

$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{IP.Setup}(1^\lambda)$; $([x], \mathsf{st}_{\mathcal{P}}) \leftarrow\!\!\$\ \mathcal{P}_{\mathsf{alg},\lambda}(\mathsf{pp})$; $H \leftarrow\!\!\$\ \Omega_{\mathsf{hLen}(\lambda)}$
$[\pi] \leftarrow\!\!\$\ \mathcal{P}^{H}_{\mathsf{alg},\lambda}(\mathsf{st}_{\mathcal{P}})$; $(a_1, c_1, \ldots, a_r, c_r, a_{r+1}) \leftarrow \pi$
$\mathsf{accept} \leftarrow (\mathsf{IP.V}(\mathsf{pp}, x, \pi) = 1) \wedge (\forall i \in [r] : c_i = H(\mathsf{pp}, x, a_1, c_1, \ldots, a_i)[:\mathsf{cLen}_i])$
$w \leftarrow\!\!\$\ \mathcal{E}(1^\lambda, \mathsf{pp}, [x], [\pi])$; Return $(\mathsf{accept} \wedge (\mathsf{pp}, x, w) \notin R)$

Fig. 4. Definition of fs-ext-1 security in the AGM. The game FS-EXT-1 defines fs-ext-1 security in the AGM for a non-uniform algebraic prover $\mathcal{P}_{\mathsf{alg}}$, an extractor $\mathcal{E}$ and a non-interactive argument obtained by applying the Fiat-Shamir transform to an interactive protocol IP. Here, IP has $r = r(\lambda)$ challenges where the i^{th} challenge is of length $\mathsf{cLen}_i = \mathsf{cLen}_i(\lambda)$ such that $\mathsf{sLen}(\lambda) \leqslant \mathsf{cLen}_i(\lambda) \leqslant \mathsf{hLen}(\lambda)$. The set $\Omega_{\mathsf{hLen}(\lambda)}$ contains all functions mapping $\{0,1\}^*$ to $\{0,1\}^{\mathsf{hLen}(\lambda)}$.

By definition, $\tau \in \mathcal{T}^{\mathsf{IP}}_{\mathsf{BadCh}}$ only if any of the challenges $c_i \in \mathsf{BadCh}(\tau')$ for some partial transcript τ' that is a prefix of τ. Now, since there are at most q queries and each of the challenges are sampled uniformly at random from Ch_i, and $|\mathsf{BadCh}(\tau')| / |\mathsf{Ch}_i| \leqslant \varepsilon$, the probability that $\tau \in \mathcal{T}^{\mathsf{IP}}_{\mathsf{BadCh}}$ is at most $q \cdot \varepsilon$.

The probability that $\tau \in \mathcal{T}^{\mathsf{IP},\mathsf{e},R}_{\mathsf{fail}}$ is $p_{\mathsf{fail}}(\mathsf{IP}, \mathcal{P}_{\mathsf{alg}}, \mathsf{e}, R, \lambda)$ in game $\mathsf{SRS}^{\mathcal{P}}_{\mathsf{IP}}$. Since $\mathcal{E}$ answers challenges honestly, the probability that $\tau \in \mathcal{T}^{\mathsf{IP},\mathsf{e},R}_{\mathsf{fail}}$ in $\mathsf{WEE\text{-}0}^{\mathcal{E},\mathcal{P}_{\mathsf{alg}},\mathcal{D}}_{\mathsf{IP},R}$ is $p_{\mathsf{fail}}(\mathsf{IP}, \mathcal{P}_{\mathsf{alg}}, \mathsf{e}, R, \lambda)$ as well.

Therefore, the probability that the output of $\mathsf{WEE\text{-}0}^{\mathcal{E},\mathcal{P}_{\mathsf{alg}},\mathcal{D}}_{\mathsf{IP},R}$ differs from the output of $\mathsf{WEE\text{-}1}^{\mathcal{P}_{\mathsf{alg}},\mathcal{D}}_{\mathsf{IP}}$ is at most $q\varepsilon + p_{\mathsf{fail}}(\mathsf{IP}, \mathcal{P}_{\mathsf{alg}}, \mathsf{e}, R, \lambda)$, i.e.,

$$\mathsf{Adv}^{\mathsf{sr\text{-}wee}}_{\mathsf{IP},R}(\mathcal{P}_{\mathsf{alg}}, \mathcal{D}, \mathcal{E}, \lambda) \leqslant q\varepsilon + p_{\mathsf{fail}}(\mathsf{IP}, \mathcal{P}_{\mathsf{alg}}, \mathsf{e}, R, \lambda) \ .$$

□

4.2 The Fiat-Shamir Transform

The Fiat-Shamir transform uses a family of hash functions $\mathcal{H}$ to convert a r-challenge public coin interactive protocol (proof or argument) IP to a non-interactive argument $\mathsf{FS}[\mathsf{IP}, \mathcal{H}]$. When $\mathcal{H}$ is modelled as a random oracle, we denote the non-interactive argument using $\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}]$. In $\mathsf{FS}[\mathsf{IP}, \mathcal{H}]$, a hash function H is first sampled from $\mathcal{H}$. A proof on public parameters pp and input x is $\pi = (a_1, c_1, a_2, c_2, \ldots, a_r, c_r, a_{r+1})$, such that

$$c_i = H(\mathsf{pp}, x, a_1, c_1, \ldots, a_{i-1}, c_{i-1}, a_i)[:\mathsf{cLen}_i]$$

for $i \in \{1, \ldots, r\}$, and IP.V returns 1 on input (pp, x, π).

FS-EXT-1 SECURITY. We formalize a notion of proof-of-knowledge (PoK) security in the AGM for non-interactive arguments obtained by applying the Fiat-Shamir transform to an interactive protocol IP. For simplicity, this notion just captures extractability instead of witness-extended emulation. We define a notion of soundness called fs-ext-1 that captures the setting where the prover has to commit to the instance beforehand. It is formally defined using the game FS-EXT-1 in Figure 4.

For an interactive proof IP and an associated relation R, algebraic prover $\mathcal{P}_{\mathsf{alg}}$, and an extractor $\mathcal{E}$, we define $\mathsf{Adv}^{\mathsf{fs\text{-}ext\text{-}1}}_{\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}],R}(\mathcal{P}_{\mathsf{alg}}, \mathcal{E}, \lambda) = \mathsf{Pr}\left[\mathsf{FS\text{-}EXT\text{-}1}^{\mathcal{P}_{\mathsf{alg}},\mathcal{E}}_{\mathsf{IP},R}(\lambda)\right]$.

The following theorem connects the online srs-wee security of a public-coin protocol IP and the fs-ext-1 soundness of non-interactive protocol $\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}]$, obtained by applying the Fiat-Shamir transform using a random oracle.

Theorem 2. *Let R be a relation. Let IP be a $r = r(\lambda)$-challenge public coin interactive protocol for the relation R where the length of the i^{th} challenge is $\mathsf{cLen}_i(\lambda)$ such that $\mathsf{sLen}(\lambda) \leqslant \mathsf{cLen}_i(\lambda) \leqslant \mathsf{hLen}(\lambda)$ for $i \in \{1, \ldots, r\}$. Let $\mathcal{E}$ be an extractor for IP. We can construct an extractor $\mathcal{E}^*$ for $\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}]$ such that for every non-uniform algebraic prover $\mathcal{P}^*_{\mathsf{alg}}$ against $\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}]$ that makes $q = q(\lambda)$ random oracle queries, there exists a non-uniform algebraic prover $\mathcal{P}_{\mathsf{alg}}$ and $\mathcal{D}$ such that for all $\lambda \in \mathbb{N}^+$*

$$\mathsf{Adv}^{\mathsf{fs\text{-}ext\text{-}1}}_{\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}],R}(\mathcal{P}^*_{\mathsf{alg}}, \mathcal{E}^*, \lambda) \leqslant \mathsf{Adv}^{\mathsf{sr\text{-}wee}}_{\mathsf{IP},R}(\mathcal{P}_{\mathsf{alg}}, \mathcal{D}, \mathcal{E}, \lambda) + \frac{q+1}{2^{\mathsf{sLen}(\lambda)}} \ .$$

Moreover, $\mathcal{P}_{\mathsf{alg}}$ makes at most q queries to its oracle and is nearly as efficient as $\mathcal{P}^*_{\mathsf{alg}}$. The extractor $\mathcal{E}^*$ is nearly as efficient as $\mathcal{E}$.

Proof. Without loss of generality we assume that $\mathcal{P}^*_{\mathsf{alg}}$ does not repeat random oracle queries. Let $r = r(\lambda)$, $\mathsf{hLen} = \mathsf{hLen}(\lambda)$, $\mathsf{sLen} = \mathsf{sLen}(\lambda)$ and $\mathsf{cLen}_i = \mathsf{cLen}_i(\lambda)$ for $i = 1, \ldots, r$. Let the length of the i^{th} prover message in IP be $l_i = l_i(\lambda)$ bits for $i \in \{1, \ldots, r+1\}$.

First we define the extractor $\mathcal{E}^*$ – it simply outputs whatever $\mathcal{E}$ returns. It follows that $\mathcal{E}^*$ is no less efficient than $\mathcal{E}$.

We set $\mathcal{D}(\cdot) = \mathsf{Acc}(\cdot)$. So, $\mathsf{Adv}^{\mathsf{sr\text{-}wee}}_{\mathsf{IP},R}(\mathcal{P}_{\mathsf{alg}}, \mathcal{D}, \mathcal{E}, \lambda)$ is essentially the probability that in $\mathsf{WEE}\text{-}0$, Acc returns $\mathtt{true}$ and $\mathcal{E}$ fails to return a valid witness.

We define adversary $\mathcal{P}_{\mathsf{alg}}$ that runs simulates the game $\mathsf{FS\text{-}EXT\text{-}1}$ to $\mathcal{P}^*_{\mathsf{alg}}$. The first stage of $\mathcal{P}_{\mathsf{alg}}$ on input pp shall run the first stage of the $\mathcal{P}^*$ on pp. If $\mathcal{P}^*$ returns $(x, \mathsf{st}_{\mathcal{P}*})$, $\mathcal{P}_{\mathsf{alg}}$ returns $(x, \mathsf{st}_{\mathcal{P}} = (\mathsf{st}_{\mathcal{P}*}, \mathsf{pp}, x))$. The second stage of $\mathcal{P}_{\mathsf{alg}}$ maintains set of states called $\mathcal{S}$ – each state is of the form $(a_1, c_1, a_2, c_2, \ldots, a_i, c_i)$. We say the length of such a state is i. On input $\mathsf{st}_{\mathcal{P}} = (\mathsf{st}_{\mathcal{P}*}, \mathsf{pp}, x)$, it first intializes $\mathcal{S}$ to $\{\varepsilon\}$ where ε is the empty string . Then it runs $\mathcal{P}^*_{\mathsf{alg}}$ on $\mathsf{st}_{\mathcal{P}*}$. It simulates the random oracle H to $\mathcal{P}^*_{\mathsf{alg}}$ as follows. On receiving a H query on y

1. $\mathcal{P}_{\mathsf{alg}}$ first checks if there exists $s \in \mathcal{S}$ of length i such that (pp, x, s) is a prefix of y i.e. $y = (\mathsf{pp}, x, s, t)$ and t is of length l_{i+1}. If the check fails, $\mathcal{P}_{\mathsf{alg}}$ returns a randomly sampled string from $\{0,1\}^{\mathsf{hLen}}$. If the check succeeds, $\mathcal{P}_{\mathsf{alg}}$ chooses the longest such state s.
2. $\mathcal{P}_{\mathsf{alg}}$ parses as y as (pp, x, s, t) and makes a query to $\mathbf{O}_{\mathrm{ext}}$ on (s, t) ans receives c as the response. $\mathcal{P}_{\mathsf{alg}}$ adds (s, t, c) to the set $\mathcal{S}$, samples a string c' from $\{0,1\}^{\mathsf{hLen}-\mathsf{cLen}_{i+1}}$ and returns (c, c').

Finally, when $\mathcal{P}^*_{\mathsf{alg}}$ returns an output π, $\mathcal{P}_{\mathsf{alg}}$ queries $\mathbf{O}_{\mathrm{ext}}$ on π and stops. It follows that $\mathcal{P}_{\mathsf{alg}}$ makes no more than q queries to its oracle and is nearly as efficient as $\mathcal{P}^*_{\mathsf{alg}}$.

Suppose the game $\mathsf{FS\text{-}EXT\text{-}1}$ returns $\mathtt{true}$. In other words $\mathcal{P}^*_{\mathsf{alg}}$ returns an accepting proof, i.e., it returns $\tau = (a_1, c_1, \ldots, a_r, c_r, a_{r+1})$ and $\mathcal{E}^*$ fails to extract a witness w.

Let $\tau_i = (a_1, c_1, \ldots, a_{i-1}, c_{i-1}, a_i)$. Now, let E be the event that $\mathcal{P}^*_{\mathsf{alg}}$ made H queries on all of $(\mathsf{pp}, x, \tau_1), \ldots, (\mathsf{pp}, x, \tau_r)$ *in order*, i.e., for all $i \in \{1, \ldots, r-1\}$, it queried $H(\mathsf{pp}, x, \tau_i)$ before $H(\mathsf{pp}, x, \tau_{i-1})$. If E happens, it is easy to see that $\mathcal{P}_{\mathsf{alg}}$ must have succeeded and $\mathcal{E}$ must have failed (since $\mathcal{E}^*$ fails only when $\mathcal{E}$ fails).

Hence, we need to upper bound the probability that τ is an accepting transcript and the event E does not happen. If τ is an accepting transcript and the event E does not happen either there exists an $i \in \{1, \ldots, r\}$ such that $H(\mathsf{pp}, x, \tau_i)$ was never queried by $\mathcal{P}^*_{\mathsf{alg}}$ or there exists $i \in \{1, \ldots, r-1\}$ such that $H(\mathsf{pp}, x, \tau_{i+1})$ was queried before $H(\mathsf{pp}, x, \tau_i)$. The probability of the former happening is at most $1/2^{\mathsf{sLen}}$ since $H(\mathsf{pp}, x, \tau_i)$ was never queried but $c_i = H(\mathsf{pp}, x, \tau_i)[:\mathsf{cLen}_i]$ is satisfied. The probability of the latter is upper bounded by the probability that a H query was made on some y before the H query on (pp, x, τ_i) such that the last $\mathsf{cLen}_i + l_{i+1}$ bits of y were (c_i, a_{i+1}). Since c_i was not fixed before the H query on (pp, x, τ_i), this happens with probability no more than $1/2^{\mathsf{sLen}}$

Game $\mathsf{FS\text{-}EXT\text{-}2}^{\mathcal{P}_{\mathsf{alg}},\mathcal{E}}_{\mathsf{IP},R}(\lambda)$:

$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{IP.Setup}(1^\lambda)$; $H \leftarrow\!\!\$\ \Omega_{\mathsf{hLen}(\lambda)}$; $([x],[\pi]) \leftarrow\!\!\$\ \mathcal{P}^{H}_{\mathsf{alg},\lambda}(\mathsf{pp})$
$(a_1, c_1, \ldots, a_r, c_r, a_{r+1}) \leftarrow \pi$
$\mathsf{accept} \leftarrow (\mathsf{IP.V}(\mathsf{pp}, x, \pi) = 1) \wedge (\forall i \in [r] : c_i = H(\mathsf{pp}, x, a_1, c_1, \ldots, a_i)[:\mathsf{cLen}_i])$
$w \leftarrow\!\!\$\ \mathcal{E}(1^\lambda, \mathsf{pp}, [x], [\pi])$; Return $(\mathsf{accept} \wedge (\mathsf{pp}, x, w) \notin R)$

Fig. 5. Definition of fs-ext-2 security in the AGM. The game FS-EXT-2 defines fs-ext-2 security in the AGM for a non-uniform algebraic prover $\mathcal{P}_{\mathsf{alg}}$, an extractor $\mathcal{E}$ and a non-interactive argument obtained by applying the Fiat-Shamir transform to an interactive protocol IP. Here, IP has $r = r(\lambda)$ challenges where the i^{th} challenge is of length $\mathsf{cLen}_i = \mathsf{cLen}_i(\lambda)$ such that $\mathsf{sLen}(\lambda) \leqslant \mathsf{cLen}_i(\lambda) \leqslant \mathsf{hLen}(\lambda)$. The set $\Omega_{\mathsf{hLen}(\lambda)}$ contains all functions mapping $\{0,1\}^*$ to $\{0,1\}^{\mathsf{hLen}(\lambda)}$.

for every query before the H query on (pp, x, τ_i). Hence, the probability that for all $i \in \{1, \ldots, r\}$, $H(\mathsf{pp}, x, \tau_i)$ was queried by $\mathcal{P}^*_{\mathsf{alg}}$ but there exists $i \in \{1, \ldots, r-1\}$ such that $H(\mathsf{pp}, x, \tau_{i+1})$ was queried before $H(\mathsf{pp}, x, \tau_i)$ is $q/2^{\mathsf{sLen}}$. Therefore, the probability that τ is an accepting transcript, but E does not happen is at most $(q+1)/2^{\mathsf{sLen}}$. Hence

$$\mathsf{Adv}^{\mathsf{fs\text{-}ext\text{-}1}}_{\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}],R}(\mathcal{P}^*_{\mathsf{alg}}, \mathcal{E}^*, \lambda) \leqslant \mathsf{Adv}^{\mathsf{sr\text{-}wee}}_{\mathsf{IP},R}(\mathcal{P}_{\mathsf{alg}}, \mathcal{D}, \mathcal{E}, \lambda) + \frac{q+1}{2^{\mathsf{sLen}(\lambda)}} \ .$$

□

In the above theorem we considered challenges in IP to be bitstrings – however, this can be adapted to protocols where the challenges are from sets that are not bitstrings. The denominator of the fraction of the bound would become the size of smallest set from which the challenges are sampled, e.g., if the challenges in the a protocol were all from the set $\mathbb{Z}^*_p$, the fraction would become $(q+1)/(p-1)$.

We can also consider an adaptive notion of soundness where the prover can output the instance and proof together- we call this notion fs-ext-2 it is formally defined using the game FS-EXT-2 in Figure 5. Unlike fs-ext-1, here the prover need not commit to the instance beforehand and can output the instance and proof together. For an interactive proof IP and an associated relation R, algebraic prover $\mathcal{P}_{\mathsf{alg}}$, and an extractor $\mathcal{E}$, we define $\mathsf{Adv}^{\mathsf{fs\text{-}ext\text{-}2}}_{\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}],R}(\mathcal{P}_{\mathsf{alg}}, \mathcal{E}, \lambda) = \mathsf{Pr}\left[\mathsf{FS\text{-}EXT\text{-}2}^{\mathcal{P}_{\mathsf{alg}},\mathcal{E}}_{\mathsf{IP},R}(\lambda)\right]$.

We assume that IP has BadCh, e functions as defined above. Further, we assume $\mathcal{T}^{\mathsf{IP}}_{\mathsf{BadCh}}$ is defined as above. We denote using $p_{\mathsf{fail},\mathsf{FS}}(\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}], \mathcal{P}_{\mathsf{alg}}, \mathsf{e}, R, \lambda)$ the probability that in $\mathsf{FS\text{-}EXT\text{-}2}^{\mathcal{P}_{\mathsf{alg}},\mathcal{E}}_{\mathsf{IP},R}$, $\mathcal{P}_{\mathsf{alg}}$ outputs $([x],[\pi])$, accept is $\mathtt{true}$, $\pi \notin \mathcal{T}^{\mathsf{IP}}_{\mathsf{BadCh}}$ but e on input $([x],[\pi])$ fails to produce a valid witness. The following theorem upper bounds the fs-ext-2 soundness of non-interactive protocol $\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}]$.

Theorem 3. *Let* IP *be an $r = r(\lambda)$-challenge public coin interactive proof for a relation R where the length of the i^{th} challenge is $\mathsf{cLen}_i(\lambda)$ such that $\mathsf{sLen}(\lambda) \leqslant \mathsf{cLen}_i(\lambda) \leqslant \mathsf{hLen}(\lambda)$ for $i \in \{1, \ldots, r\}$. Assume that* BadCh *and* e *are as given above. Let τ' be a partial transcript such that the challenge that comes right after is sampled from Ch_i. Assume that for all $i \in \{1, \ldots, r\}$, we have that $|\mathsf{BadCh}(\tau')| / |\mathsf{Ch}_i| \leqslant \varepsilon$ for some $\varepsilon \in [0,1]$. Then, there exists an extractor $\mathcal{E}^*$ that uses* e *such that for any non-uniform algebraic prover $\mathcal{P}^*_{\mathsf{alg}}$ for $\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}]$ making at most $q = q(\lambda)$ queries to its random oracle, for all $\lambda \in \mathbb{N}^+$*

$$\mathsf{Adv}^{\mathsf{fs\text{-}ext\text{-}2}}_{\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}],R}(\mathcal{P}^*_{\mathsf{alg}}, \mathcal{E}^*, \lambda) \leqslant q\varepsilon + p_{\mathsf{fail},\mathsf{FS}}(\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}], \mathcal{P}^*_{\mathsf{alg}}, \mathsf{e}, R, \lambda) \ .$$

The time complexity of the extractor $\mathcal{E}^*$ is $O(q \cdot t_V + t_{\mathsf{e}})$ where t_V is the time required to run IP.V *and t_{e} is the time required to run* e.

$\mathsf{InPrd.P}(((n, \mathbf{g}, \mathbf{h}, u), P), (\mathbf{a}, \mathbf{b}))$

$\mathbf{g}^{(0)} \leftarrow \mathbf{g}; \mathbf{h}^{(0)} \leftarrow \mathbf{h}$

$n_0 \leftarrow n; P^{(0)} \leftarrow P; \mathbf{a}^{(0)} \leftarrow \mathbf{a}; \mathbf{b}^{(0)} \leftarrow \mathbf{b}$

For $i = 1, \ldots, \log n$

$n_i \leftarrow n_{i-1}/2$

$c_L \leftarrow \langle \mathbf{a}^{(i)}[: n_i], \mathbf{b}^{(i)}[n_i :] \rangle$

$c_R \leftarrow \langle \mathbf{a}^{(i)}[n_i :], \mathbf{b}^{(i)}[: n_i] \rangle$

$L_i \leftarrow \left(\mathbf{g}^{(i-1)}_{[n_i:]}\right)^{\mathbf{a}^{(i)}[:n_i]} \left(\mathbf{h}^{(i-1)}_{[:n_i]}\right)^{\mathbf{b}^{(i)}[n_i:]} u^{c_L}$

$R_i \leftarrow \left(\mathbf{g}^{(i-1)}_{[:n_i]}\right)^{\mathbf{a}^{(i)}[n_i:]} \left(\mathbf{h}^{(i-1)}_{[n_i:]}\right)^{\mathbf{b}^{(i)}[:n_i]} u^{c_R}$ $\xrightarrow{L_i, R_i}$

$\xleftarrow{x_i}$ $x_i \leftarrow_{\$} \mathbb{Z}_p^*$

$\mathbf{g}^{(i)} \leftarrow \left(\mathbf{g}^{(i-1)}_{[:n_i]}\right)^{x_i^{-1}} \circ \left(\mathbf{g}^{(i-1)}_{[n_i:]}\right)^{x_i}$

$\mathbf{h}^{(i)} \leftarrow \left(\mathbf{h}^{(i-1)}_{[:n_i]}\right)^{x_i} \circ \left(\mathbf{h}^{(i-1)}_{[n_i:]}\right)^{x_i^{-1}}$

$P^{(i)} \leftarrow L_i^{x_i^2} P^{(i-1)} R_i^{x_i^{-2}}$

$\mathbf{a}^{(i)} \leftarrow \mathbf{a}^{(i-1)}[: n_i] x^{-1} + \mathbf{a}^{(i)}[n_i :] x$

$\mathbf{b}^{(i)} \leftarrow \mathbf{b}^{(i-1)}[: n_i] x + \mathbf{b}^{(i)}[n_i :] x^{-1}$

$g \leftarrow \mathbf{g}^{(\log n)}; h \leftarrow \mathbf{h}^{(\log n)}$

$a \leftarrow \mathbf{a}^{(\log n)}; b \leftarrow \mathbf{b}^{(\log n)}$

$\xrightarrow{a, b}$

$\mathsf{InPrd.V}((n, \mathbf{g}, \mathbf{h}, u), P)$

$\mathbf{g}^{(0)} \leftarrow \mathbf{g}; \mathbf{h}^{(0)} \leftarrow \mathbf{h}$

$n_0 \leftarrow n; P^{(0)} \leftarrow P$

For $i = 1, \ldots, \log n$

$n_i \leftarrow n_{i-1}/2$

$\mathbf{g}^{(i)} \leftarrow \left(\mathbf{g}^{(i-1)}_{[:n_i]}\right)^{x_i^{-1}} \circ \left(\mathbf{g}^{(i-1)}_{[n_i:]}\right)^{x_i}$

$\mathbf{h}^{(i)} \leftarrow \left(\mathbf{h}^{(i-1)}_{[:n_i]}\right)^{x_i} \circ \left(\mathbf{h}^{(i-1)}_{[n_i:]}\right)^{x_i^{-1}}$

$P^{(i)} \leftarrow L_i^{x_i^2} P^{(i-1)} R_i^{x_i^{-2}}$

$g \leftarrow \mathbf{g}^{(\log n)}; h \leftarrow \mathbf{h}^{(\log n)}$

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

Fig. 6. Bulletproofs inner-product argument InPrd.

The proof of this theorem is similar to Theorem 1 and has been omitted.

5 Online srs-wee Security of Bulletproofs

In this section, we shall apply our framework to prove online srs-wee security in the AGM for two instantiations of Bulletproofs- range proofs (RngPf) and proofs for arithmetic circuit satisfiability (ACSPf). We first introduce the Bulletproofs inner product argument (InPrd) in Section 5.1 which forms the core of both RngPf and ACSPf. Then, in Sections 5.2 and 5.3 we introduce and analyze online srs-wee security of RngPf and ACSPf respectively.

5.1 Inner Product Argument InPrd

We shall assume that $\mathsf{InPrd} = \mathsf{InPrd}[\mathbb{G}]$ is instantiated on an understood family of groups $\mathbb{G} = \{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}^+}$ of order $p = p(\lambda)$. Using InPrd, a prover can convince a verifier that $P \in \mathbb{G}$ is a well-formed commitment to vectors $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n$ and their inner-product $\langle \mathbf{a}, \mathbf{b} \rangle$. More precisely, the prover wants to prove to the verifier that $P = \mathbf{g}^{\mathbf{a}} \mathbf{h}^{\mathbf{b}} u^{\langle \mathbf{a}, \mathbf{b} \rangle}$ where $\mathbf{g} \in \mathbb{G}^n, \mathbf{h} \in \mathbb{G}^n, u \in \mathbb{G}$ are independent generators of $\mathbb{G}$. We assume that n is a power of 2 without loss of generality since if needed, one can pad the input appropriately to ensure that this holds. The prover and the verifier for InPrd is formally defined in Figure 6.

5.2 Online srs-wee Security of RngPf

We shall assume that $\mathsf{RngPf} = \mathsf{RngPf}[\mathbb{G}]$ is instantiated on an understood family of groups $\mathbb{G} = \{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}^+}$ of order $p = p(\lambda)$. The argument RngPf is an argument of knowledge for the relation

$$R = \left\{ \left((n \in \mathbb{N}, g, h \in \mathbb{G}), V \in \mathbb{G}, (v, \gamma \in \mathbb{Z}_p) \right) : g^v h^\gamma = V \wedge v \in [0, 2^n - 1] \right\}. \tag{4}$$

$\mathsf{RngPf.P}((n, \mathbf{g}, \mathbf{h}, g, h, u), V), (v, \gamma))$ | | $\mathsf{RngPf.V}((n, \mathbf{g}, \mathbf{h}, g, h, u), V)$

$\mathbf{a}_L \leftarrow \mathsf{BinRep}(v, n); \mathbf{a}_R \leftarrow \mathbf{a}_L - \mathbf{1}^n$

$\alpha \leftarrow\!\!\$\ \mathbb{Z}_p; A \leftarrow h^\alpha \mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R}$

$\mathbf{s}_L \leftarrow\!\!\$\ \mathbb{Z}_p^n; \mathbf{s}_R \leftarrow\!\!\$\ \mathbb{Z}_p^n$

$\rho \leftarrow\!\!\$\ \mathbb{Z}_p; S \leftarrow h^\rho \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R}$ | $\xrightarrow{A,S}$ |

| $\xleftarrow{y,z}$ | $y, z \leftarrow\!\!\$\ \mathbb{Z}_p^*$

$l(X) \leftarrow (\mathbf{a}_L - z \cdot \mathbf{1}^n) + \mathbf{s}_L \cdot X$ | | $\delta(y, z) \leftarrow (z - z^2) \cdot \langle \mathbf{1}^n, \mathbf{y}^n \rangle + z^2 \cdot \mathbf{2}^n$

$r(X) \leftarrow \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n + \mathbf{s}_R \cdot X) + z^2 \cdot \mathbf{2}^n$

$t(X) \leftarrow \langle l(X), r(X) \rangle = t_0 + t_1 X + t_2 X^2$

$\beta_1, \beta_2 \leftarrow\!\!\$\ \mathbb{Z}_p$

$T_i \leftarrow g^{t_i} h^{\beta_i}$ for $i \in \{1, 2\}$ | $\xrightarrow{T_1,T_2}$ |

| $\xleftarrow{x}$ | $x \leftarrow\!\!\$\ \mathbb{Z}_p^*$

$\mathbf{l} \leftarrow l(x); \mathbf{r} \leftarrow r(x); \hat{t} \leftarrow \langle \mathbf{l}, \mathbf{r} \rangle$

$\beta_x \leftarrow \beta_2 \cdot x^2 + \beta_1 \cdot x + z^2 \gamma; \mu \leftarrow \alpha + \rho \cdot x$ | $\xrightarrow{\beta_x, \mu, \hat{t}}$ |

| $\xleftarrow{w}$ | $w \leftarrow\!\!\$\ \mathbb{Z}_p^*$

$\mathbf{h}' \leftarrow \mathbf{h}^{\mathbf{y}^{-n}}; u' \leftarrow u^w$ | | $\mathbf{h}' \leftarrow \mathbf{h}^{\mathbf{y}^{-n}}; u' \leftarrow u^w$

$P \leftarrow A S^x \mathbf{g}^{-z \cdot \mathbf{1}^n} \mathbf{h}'^{z \cdot \mathbf{y}^n + z^2 \cdot \mathbf{2}^n}$ | | $P \leftarrow A S^x \mathbf{g}^{-z \cdot \mathbf{1}^n} \mathbf{h}'^{z \cdot \mathbf{y}^n + z^2 \cdot \mathbf{2}^n}$

$P' \leftarrow h^{-\mu} P (u')^{\hat{t}}$ | | $P' \leftarrow h^{-\mu} P (u')^{\hat{t}}$

$\mathsf{InPrd.P}((\mathbf{g}, \mathbf{h}', u', P'), (\mathbf{l}, \mathbf{r}))$ | $\Longleftrightarrow$ | $\mathsf{InPrd.V}(\mathbf{g}, \mathbf{h}', u', P') \rightarrow b$

$R \leftarrow V^{z^2} g^{\delta(y,z)} T_1^x T_2^{x^2}$

If $b = 1 \wedge g^{\hat{t}} h^{\beta_x} = R$ then

  Return 1

Return 0

Fig. 7. Prover and Verifier for RngPf. The function $\mathsf{BinRep}(v, n)$ outputs the n-bit representation of v. The symbol $\Longleftrightarrow$ denotes the interaction between InPrd.P and InPrd.V with the output of the InPrd.V being b.

DESCRIPTION OF RngPf. RngPf.Setup returns $\mathbf{g} \in \mathbb{G}^n, \mathbf{h} \in \mathbb{G}^n, g, h, u \in \mathbb{G}$ where $\mathbf{g}, \mathbf{h}$ are vectors of independent generators and g, h, u are other independent generators of the group $\mathbb{G}$. The prover and verifier for RngPf are defined in Figure 7.

In Theorem 4, we analyze the online srs-wee security for RngPf. Since RngPf has a group element V in its input, the analysis of non-adaptive srs-wee security would differ from the online srs-wee analysis. In Section 7, we analyse the non-adaptive srs-wee security of RngPf – it turns out that the proof is even harder for this case because the function e does not have the representation of V. The resulting bound is increased to the square root of the adaptive bound, due to our limited use of rewinding.

Theorem 4. *Let $\mathbb{G} = \{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}^+}$ be a family of groups of order $p = p(\lambda)$. Let $\mathsf{RngPf} = \mathsf{RngPf}[\mathbb{G}]$ be the interactive argument as defined in Figure 7, for the relation R in (4). We can construct an extractor $\mathcal{E}$ such that for any non-uniform algebraic prover $\mathcal{P}_{\mathsf{alg}}$ making at most $q = q(\lambda)$ queries to its oracle, there exists a non-uniform adversary $\mathcal{F}$ with the property that for any (computationally unbounded) distinguisher $\mathcal{D}$, for all $\lambda \in \mathbb{N}^+$*

$$\mathsf{Adv}^{\mathsf{sr-wee}}_{\mathsf{RngPf},R}(\mathcal{P}_{\mathsf{alg}}, \mathcal{D}, \mathcal{E}, \lambda) \leqslant \frac{(14n + 8)q}{p - 1} + \mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{F}, \lambda) + \frac{1}{p} .$$

Moreover, the time complexity of the extractor $\mathcal{E}$ is $O(q \cdot n)$ and that of adversary $\mathcal{F}$ is $O(q \cdot n)$.

We show that the bound above is tight in Theorem 5. Using Theorem 2, we get the following corollary.

Corollary 1. *Let $\mathbb{G} = \{\mathbb{G}_\lambda\}_{\lambda\in\mathbb{N}^+}$ be a family of groups of order $p = p(\lambda)$. Let $\mathsf{RngPf} = \mathsf{RngPf}[\mathbb{G}]$ be the interactive argument as defined in Figure 7, for the relation R in (4). Let $\mathsf{FS}^{\mathbf{RO}}[\mathsf{RngPf}]$ be the non-interactive argument obtained by applying the Fiat-Shamir transform to RngPf using a random oracle. We can construct an extractor $\mathcal{E}$ such that for any non-uniform algebraic prover $\mathcal{P}_{\mathsf{alg}}$ making at most $q = q(\lambda)$ queries to the random oracle there exists a non-uniform adversary $\mathcal{F}$ with the property that for all $\lambda \in \mathbb{N}^+$*

$$\mathsf{Adv}^{\mathsf{fs\text{-}ext\text{-}1}}_{\mathsf{FS}^{\mathbf{RO}}[\mathsf{RngPf}],R}(\mathcal{P}_{\mathsf{alg}},\mathcal{E},\lambda) \leqslant \frac{(14n+9)q+1}{p-1} + \mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{F},\lambda) + \frac{1}{p}\,.$$

Moreover, the time complexity of the extractor $\mathcal{E}$ is $O(q\cdot n)$ and that of adversary $\mathcal{F}$ is $O(q\cdot n)$.

Proof (Theorem 4). In order to prove this theorem, we invoke Theorem 1 by defining BadCh and e and showing that $\varepsilon \leqslant \frac{14n+8}{p-1}$ and there exists an adversary $\mathcal{F}$ such that $p_{\mathsf{fail}}(\mathsf{RngPf},\mathcal{P}_{\mathsf{alg}},\mathsf{e},R,\lambda) \leqslant \mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{F}) + \frac{1}{p}$.

DEFINING BadCh AND UPPER BOUNDING ε. To start off, we define $\mathsf{BadCh}(\tau')$ for all partial transcripts τ'. Let Ch be the set from which the challenge that just follows τ' is sampled. We use a helper function $\mathsf{CheckBad}$ to define $\mathsf{BadCh}(\tau')$. The function $\mathsf{CheckBad}$ takes as input a partial extended transcript $[\tau']$ and a challenge $c \in \mathsf{Ch}$ and returns $\mathtt{true}$ if and only if $c \in \mathsf{BadCh}(\tau')$. For each verifier challenge in RngPf, there is a definition of $\mathsf{CheckBad}$ in Figure 8. Every $\mathsf{CheckBad}$ function defines several bad conditions that depend on τ' – most of these bad conditions are checked using the predicate SZ. This predicate takes as input a vector of polynomials and a corresponding vector of points to evaluate the polynomial on and returns $\mathtt{true}$ iff any of the polynomials is non-zero but its evaluation at the corresponding point is zero. One can safely ignore the details of the definitions of $\mathsf{CheckBad}$ functions for now – the rationale behind their definitions shall become apparent later on.

The following lemma establishes an upper bound of $(14n+8)/(p-1)$ on $|\mathsf{BadCh}(\tau')|/|\mathsf{Ch}|$.

Lemma 5. *Let τ' be a partial transcript for RngPf. Let Ch be the set from which the challenge that comes right after τ' is sampled. Then, $\frac{|\mathsf{BadCh}(\tau')|}{|\mathsf{Ch}|} \leqslant \frac{14n+8}{p-1}$.*

Proof. The proof of this lemma proceeds by computing an upper bound on the maximum fraction of c's in Ch for which $\mathsf{CheckBad}(\tau',c)$ will return $\mathtt{true}$, for all the definitions of $\mathsf{CheckBad}$, using the Schwartz-Zippel Lemma.

The function $\mathsf{CheckBad}(\tau',(y,z))$ returns $\mathtt{true}$ if $\mathsf{SZ}(f(Y,Z),(y,z))$ is $\mathtt{true}$. The polynomial $f(Y,Z)$ is a polynomial of degree at most $n+1$. So, the fraction of (y,z)'s for which $\mathsf{SZ}(f(Y,Z),(y,z))$ is $\mathtt{true}$ is at most $(n+1)/(p-1)$ using the Schwartz-Zippel Lemma. Hence, the fraction of $y,z \in \mathbb{Z}_p^*$ for which $\mathsf{CheckBad}(\tau',(y,z))$ returns $\mathtt{true}$ is at most $(n+1)/(p-1)$.

The function $\mathsf{CheckBad}(\tau',x)$ returns $\mathtt{true}$ if any of $\mathsf{SZ}(f_i(X),x)$ for $i = 1,2,3,4$ is $\mathtt{true}$. Since $f_1(X)$ and $f_2(X)$ are vectors of n polynomials, each polynomial of degree 2, we get that the fraction of x's in $\mathbb{Z}_p^*$ for which $\mathsf{SZ}(f_i(X),x)$ is $\mathtt{true}$ for $i = 1,2$ is at most $2n/(p-1)$. The polynomials $f_3(X), f_4(X)$ are polynomials of degree at most 2. The fraction of x's in $\mathbb{Z}_p^*$ for which $\mathsf{SZ}(f_3(X),x)$ or $\mathsf{SZ}(f_4(X),x)$ is $\mathtt{true}$ is at most $2/(p-1)$. Using the union bound, the fraction of x's in $\mathbb{Z}_p^*$ such that $\mathsf{CheckBad}(\tau',x)$ returns $\mathtt{true}$ is at most $(4n+4)/(p-1)$.

The function $\mathsf{CheckBad}(\tau',w)$ returns $\mathtt{true}$ if $\mathsf{SZ}(f(W),w)$ is $\mathtt{true}$. The polynomial $f(W)$ is a polynomial of degree 1, hence using the Schwartz-Zippel Lemma the fraction of w's in $\mathbb{Z}_p^*$ for which $\mathsf{CheckBad}(\tau',w)$ returns $\mathtt{true}$ is at most $1/(p-1)$.

Procedure $\mathsf{CheckBad}(\lceil\tau'\rceil, (y,z))$:
$//\lceil\tau'\rceil = \big((n,\mathbf{g},\mathbf{h},u,g,h),[V],([A],[S])\big)$
$f(Y,Z) \leftarrow Z^2(v_g - \langle a_{\mathbf{g}}, \mathbf{2}^n\rangle) - Z\langle a_{\mathbf{g}} - a_{\mathbf{h}} - \mathbf{1}^n, \mathbf{Y}^n\rangle - \langle a_{\mathbf{g}} \circ a_{\mathbf{h}}, \mathbf{Y}^n\rangle$
Return $\mathsf{SZ}(f(Y,Z),(y,z))$

Procedure $\mathsf{CheckBad}(\lceil\tau'\rceil, x)$:
$//\lceil\tau'\rceil = \big((n,\mathbf{g},\mathbf{h},u,g,h),[V],([A],[S]),(y,z),([T_1],[T_2])\big)$
$f_1(X) \leftarrow v_{\mathbf{g}}z^2 + t_{1\mathbf{g}}X + t_{2\mathbf{g}}X^2$; $f_2(X) \leftarrow v_{\mathbf{h}}z^2 + t_{1\mathbf{h}}X + t_{2\mathbf{h}}X^2$
$f_3(X) \leftarrow v_u z^2 + t_{1u}X + t_{2u}X^2$; $\delta(y,z) \leftarrow (z-z^2)\langle \mathbf{1}^n, \mathbf{y}^n\rangle - z^3\langle \mathbf{1}^n, \mathbf{2}^n\rangle$
$l(X) \leftarrow (a_{\mathbf{g}} - z\cdot\mathbf{1}^n) + s_{\mathbf{g}}\cdot X$; $r(X) \leftarrow \mathbf{y}^n \circ (a_{\mathbf{h}} + z\cdot\mathbf{1}^n + s_{\mathbf{h}}\cdot X) + z^2\cdot\mathbf{2}^n$; $f_4(X) \leftarrow v_g z^2 + \delta(y,z) + t_{1g}X + t_{2g}X^2 - \langle l(X), r(X)\rangle$
Return $\mathsf{SZ}(f_1(X),x) \vee \mathsf{SZ}(f_2(X),x) \vee \mathsf{SZ}(f_3(X),x) \vee \mathsf{SZ}(f_4(X),x)$

Procedure $\mathsf{CheckBad}(\lceil\tau'\rceil, w)$:
$//\lceil\tau'\rceil = \big((n,\mathbf{g},\mathbf{h},u,g,h),[V],([A],[S]),(y,z),([T_1],[T_2]),x,(\beta_x,\mu,\hat{t})\big)$
$\mathbf{l} \leftarrow (a_{\mathbf{g}} - z\cdot\mathbf{1}^n) + s_{\mathbf{g}}\cdot x$; $\mathbf{r} \leftarrow (a_{\mathbf{h}} + xs_{\mathbf{h}} + z\mathbf{1}^n) \circ \mathbf{y}^n + z^2\mathbf{2}^n$; $f(W) \leftarrow W\hat{t} - W\langle \mathbf{l}, \mathbf{r}\rangle$
Return $\mathsf{SZ}(f(W), w)$

Procedure $\mathsf{CheckBad}(\lceil\tau'\rceil, x_m)$:
$//\lceil\tau'\rceil = \big((n,\mathbf{g},\mathbf{h},u,g,h),[V],([A],[S]),(y,z),([T_1],[T_2]),x,(\beta_x,\mu,\hat{t}),w,([L_1],[R_1]),x_1,\ldots,([L_m],[R_m])\big)$
$p'_{\mathbf{g}} \leftarrow a_{\mathbf{g}} + xs_{\mathbf{g}} - z\mathbf{1}^n$; $p'_{\mathbf{h}} \leftarrow a_{\mathbf{h}} + xs_{\mathbf{h}} + \mathbf{y}^{-n} \circ (z\mathbf{y}^n + z^2\mathbf{2}^n)$; $p'_u \leftarrow a_u + xs_u + w\hat{t}$
For $j = 0,\ldots,n-1$ do
  $f^{\mathbf{g}}_{m,j}(X) \leftarrow l_{mg_{1+j}} + X^2 + r_{mg_{1+j}}X^{-2} + p'_{g_{1+j}} + \sum_{i=1}^{m-1}(l_{ig_{1+j}}x_i^2 + r_{ig_{1+j}}x_i^{-2})$
  $f^{\mathbf{h}}_{m,j}(X) \leftarrow l_{mh_{1+j}}X^2 + r_{mh_{1+j}}X^{-2} + p'_{h_{1+j}} + \sum_{i=1}^{m-1}(l_{ih_{1+j}}x_i^2 + r_{ih_{1+j}}x_i^{-2})$
$f^u_m(X) \leftarrow l_{mu}X^2 + r_{mu}X^{-2} + p'_u + \sum_{i=1}^{m-1}(l_{iu}x_i^2 + r_{iu}x_i^{-2})$
flag $\leftarrow$ `false`
For $t = 1,\ldots,m-1$ do for $j = 0,\ldots,n/2^t - 1$ do
  flag $\leftarrow$ flag $\vee$ $\mathsf{SZ}(f^{\mathbf{g}}_{m,j}(X)\cdot x_t^2 - f^{\mathbf{g}}_{m,j+n/2^t}(X), x_m) \vee \mathsf{SZ}(f^{\mathbf{h}}_{m,j}(X) - f^{\mathbf{h}}_{m,j+n/2^t}(X)\cdot x_t^2, x_m)$
For $j = 0,\ldots,n/2^m - 1$ do
  flag $\leftarrow$ flag $\vee$ $\mathsf{SZ}(f^{\mathbf{g}}_{m,j}(X)\cdot X^2 - f^{\mathbf{g}}_{m,j+n/2^m}(X), x_m) \vee \mathsf{SZ}(f^{\mathbf{h}}_{m,j}(X) - f^{\mathbf{h}}_{m,j+n/2^m}(X)\cdot X^2, x_m)$
flag $\leftarrow$ flag $\vee$ $\mathsf{SZ}\Big(f^u_m(X) - w\cdot\sum_{j=0}^{n/2^m-1} f^{\mathbf{g}}_{m,j}(X)\cdot f^{\mathbf{h}}_{m,j}(X)\cdot y^j, x_m\Big)$
Return flag

Fig. 8. The functions CheckBad function for the RngPf.

The function $\mathsf{CheckBad}(\tau', x_m)$ returns `true` if and only if SZ is `true` for any of the $\sum_{t=1}^{m-1} 2n/2^t$ polynomials of degree at most 4 (the degree here is the difference between highest and lowest degree), $2n/2^m$ polynomials of degree at most 6 and one polynomial of degree at most 8. Using Schwartz Zippel Lemma and the union bound the fraction of x_m's for which $\mathsf{CheckBad}(\tau', x_m)$ returns `true` is at most

$$\frac{8}{p-1}\left(\sum_{t=1}^{m-1}\frac{n}{2^t}\right) + \frac{12n}{2^m(p-1)} + \frac{8}{p-1}\ .$$

This fraction is at most $(14n+8)/(p-1)$ for $m \in \{1,\ldots,\log n\}$.

Therefore the maximum value of $|\mathsf{BadCh}(\tau')|/|\mathsf{Ch}|$ for any partial transcript τ', i.e., the maximum fraction of c's for which $\mathsf{CheckBad}(\tau', c)$ is `true` is upper bounded by $(14n+8)/(p-1)$. □

DEFINING e. Let τ be a transcript of RngPf as defined below.

$$\begin{aligned}\tau = \big((n,\mathbf{g},\mathbf{h},u,g,h), V; (A,S), (y,z), (T_1,T_2), x, (\beta_x,\mu,\hat{t}), w, (L_1,R_1), x_1, (L_2,R_2), x_2, \ldots,\\ (L_{\log n}, R_{\log n}), x_{\log n}, (a,b)\big)\ .\end{aligned} \tag{5}$$

Procedure $\mathsf{e}([\tau])$:

$//[\tau] = \big((n, \mathbf{g}, \mathbf{h}, u, g, h), [V]\,; ([A]\,, [S]), (y, z), ([T_1]\,, [T_2]), x, (\beta_x, \mu, \hat{t}), w, ([L_1]\,, [R_1]), x_1, \ldots, ([L_{\log n}]\,, [R_{\log n}]), x_{\log n}, (a, b)\big)$

$v^* \leftarrow v_g$; $\gamma^* \leftarrow v_h$; Return (v^*, γ^*)

Fig. 9. The function e for RngPf.

Let us represent using $\tau|_c$ the prefix of τ just before the challenge c. For example

$$\tau|_{(y,z)} = \big((n, \mathbf{g}, \mathbf{h}, u, g, h), V, (A, S)\big)\ .$$

The function e simply returns (v_g, v_h). However, its output is a valid witness only if $v_{\mathbf{g}} = v_{\mathbf{h}} = \mathbf{0}^n, v_u = 0$ and $v_g \in [0, 2^n - 1]$.

PROVING AN UPPER BOUND ON $p_{\mathsf{fail}}(\mathsf{RngPf}, \mathcal{P}_{\mathsf{alg}}, \mathsf{e}, R, \lambda)$. We construct an adversary $\mathcal{H}$ against the discrete logarithm relation problem that takes as input independent generators $\mathbf{g}, \mathbf{h}, g, h, u$ of the group $\mathbb{G}$ and works as follows. It simulates the game $\mathsf{SRS}_{\mathsf{RngPf}}$ to $\mathcal{P}_{\mathsf{alg}}$ using public parameters $n, \mathbf{g}, \mathbf{h}, g, h, u$. If $\mathcal{P}_{\mathsf{alg}}$ manages to produce an accepting transcript τ, $\mathcal{H}$ calls a helper function h on input $[\tau]$ and outputs whatever h outputs. We shall define h in such a way that for $\tau \notin \mathcal{T}^{\mathsf{RngPf}}_{\mathsf{BadCh}}$ if $\mathsf{h}([\tau])$ returns a trivial discrete logarithm relation, then $\mathsf{e}([\tau])$ returns a valid witness. In other words, we have that whenever $\mathsf{e}([\tau])$ fails to extract a valid witness for an accepting transcript $\tau \notin \mathcal{T}^{\mathsf{RngPf}}_{\mathsf{BadCh}}$, $\mathsf{h}([\tau])$ outputs a non-trivial discrete logarithm relation, i.e., $\mathcal{H}$ succeeds. So we have that $p_{\mathsf{fail}}(\mathsf{RngPf}, \mathcal{P}_{\mathsf{alg}}, \mathsf{e}, R, \lambda) \leqslant \mathsf{Adv}^{\mathsf{dl\text{-}rel}}_{\mathbb{G}, 2n+3}(\mathcal{H})$. Using Lemma 2 we would have that there exists an adversary $\mathcal{F}$ such that $p_{\mathsf{fail}}(\mathsf{RngPf}, \mathcal{P}_{\mathsf{alg}}, \mathsf{e}, R, \lambda) \leqslant \mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{F}) + \frac{1}{p}$. We also have that $\mathcal{F}$ is nearly as efficient as $\mathcal{H}$.

DEFINING h. We next describe the h function. Let τ, as defined in (5), be an accepting transcript. $V^{z^2} g^{\delta(y,z)} T_1^x T_2^{x^2} = g^{\hat{t}} h^{\beta_x}$. must hold since τ is an accepting transcript.

The function h can plug in the representations of T_1, T_2, V into the above equation and compute $e^{(1)}_{\mathbf{g}}, e^{(1)}_{\mathbf{h}}, e^{(1)}_g, e^{(1)}_h, e^{(1)}_u$ such that $\mathbf{g}^{e^{(1)}_{\mathbf{g}}} \mathbf{h}^{e^{(1)}_{\mathbf{h}}} g^{e^{(1)}_g} h^{e^{(1)}_h} u^{e^{(1)}_u} = 1$. If not all of these are zero, h returns $e^{(1)}_{\mathbf{g}}, e^{(1)}_{\mathbf{h}}, e^{(1)}_g, e^{(1)}_h, e^{(1)}_u$.

Again since τ is an accepting transcript, $\mathsf{InPrd.V}$ must have returned 1 and hence $P^{(\log n)} = (\mathbf{g}^{(\log n)})^a (\mathbf{h}^{(\log n)})^b u^{ab}$ must hold. All the terms in the above equality can be expressed in terms of $\mathbf{g}, \mathbf{h}, g, h, u$ and one can compute $e^{(2)}_{\mathbf{g}}, e^{(2)}_{\mathbf{h}}, e^{(2)}_g, e^{(2)}_h, e^{(2)}_u$ such that $\mathbf{g}^{e^{(2)}_{\mathbf{g}}} \mathbf{h}^{e^{(2)}_{\mathbf{h}}} g^{e^{(2)}_g} h^{e^{(2)}_h} u^{e^{(2)}_u} = 1$. The function h computes and returns $e^{(2)}_{\mathbf{g}}, e^{(2)}_{\mathbf{h}}, e^{(2)}_g, e^{(2)}_h, e^{(2)}_u$. We define the function h formally in Figure 10. It follows from the description of h that it runs in time $O(n)$. The running time of $\mathcal{H}$ consists of the time required to answers q queries, run $\mathsf{RngPf.V}$ in at most q paths in the execution tree and the time required to run h. Hence its time complexity is $O(q \cdot n)$. Using Lemma 2, time complexity of $\mathcal{F}$ is $O(q \cdot n)$.

RELATING h, e. In order to complete the proof of Theorem 4, in the following lemma we show that – for an accepting transcript τ such that $\tau \notin \mathcal{T}^{\mathsf{RngPf}}_{\mathsf{BadCh}}$ if $\mathsf{h}([\tau])$ returns a trivial discrete logarithm relation, then $\mathsf{e}([\tau])$ returns a valid witness.

Lemma 6. *Let τ, as defined in (5), be an accepting transcript of* RngPf *such that $\tau \notin \mathcal{T}^{\mathsf{RngPf}}_{\mathsf{BadCh}}$. If $\mathsf{h}([\tau])$ returns $(\mathbf{0}^n, \mathbf{0}^n, 0, 0, 0)$ then $\mathsf{e}([\tau])$ returns (v^*, γ^*) such that*

$$g^{v^*} h^{\gamma^*} = V \text{ and } v^* \in [0, 2^n - 1]\ .$$

Procedure $\mathsf{h}([\tau])$:

$//[\tau] = \big((n, \mathbf{g}, \mathbf{h}, u, g, h), [V]\,; ([A]\,, [S]), (y, z), ([T_1]\,, [T_2]), x, (\beta_x, \mu, \hat{t}), w, ([L_1]\,, [R_1]), x_1, \ldots,$
$([L_{\log n}]\,, [R_{\log n}]), x_{\log n}, (a, b)\big)$

$\delta(y,z) \leftarrow (z - z^2)\langle \mathbf{1}^n, \mathbf{y}^n\rangle - z^3\langle \mathbf{1}^n, \mathbf{2}^n\rangle$

$e_{\mathbf{g}}^{(1)} \leftarrow v_{\mathbf{g}} z^2 + t_{1\mathbf{g}} x + t_{2\mathbf{g}} x^2$; $e_{\mathbf{h}}^{(1)} \leftarrow v_{\mathbf{h}} z^2 + t_{1\mathbf{h}} x + t_{2\mathbf{h}} x^2$; $e_u^{(1)} \leftarrow v_u z^2 + t_{1u} x + t_{2u} x^2$; $e_g^{(1)} \leftarrow v_g z^2 + \delta(y,z) + t_{1g} x + t_{2g} x^2 - \hat{t}$;
$e_h^{(1)} \leftarrow v_h z^2 + t_{1h} x + t_{2h} x^2 - \beta_x$

If $(e_{\mathbf{g}}^{(1)}, e_{\mathbf{h}}^{(1)}, e_u^{(1)}, e_g^{(1)}, e_h^{(1)}) \neq (\mathbf{0}^n, \mathbf{0}^n, 0, 0, 0)$ then return $(e_{\mathbf{g}}^{(1)}, e_{\mathbf{h}}^{(1)}, e_u^{(1)}, e_g^{(1)}, e_h^{(1)})$

$p'_{\mathbf{g}} \leftarrow (a_{\mathbf{g}}) + x s_{\mathbf{g}} - z\mathbf{1}^n$; $p'_{\mathbf{h}} \leftarrow a_{\mathbf{h}} + x s_{\mathbf{h}} + \mathbf{y}^{-n} \circ (z\mathbf{y}^n + z^2 \mathbf{2}^n)$

$p'_g \leftarrow a_g + x s_g$; $p'_h \leftarrow a_h + x s_h - \mu$; $p'_u \leftarrow a_u + x s_u + w\hat{t}$

For $k = 0$ to $n - 1$ do

$e_{g_{k+1}}^{(2)} \leftarrow p'_{g_{1+k}} + \sum_{i=1}^{\log n} l_{ig_{1+k}} x_i^2 + r_{ig_{1+k}} x_i^{-2} - a \cdot \prod_{i=1}^{\log n} x_i^{(-1)^{1-\mathsf{bit}(k,i,\log n)}}$

$e_{h_{k+1}}^{(2)} \leftarrow p'_{h_{1+k}} + \sum_{i=1}^{\log n} l_{ih_{1+k}} x_i^2 + r_{ih_{1+k}} + x_i^{-2} - b y^{(-(k))} \cdot \prod_{i=1}^{\log n} x_i^{(-1)^{\mathsf{bit}(k,i,\log n)}}$

$e_{\mathbf{g}}^{(2)} \leftarrow (e_{g_1}^{(2)}, \ldots, e_{g_n}^{(2)})$; $e_{\mathbf{h}}^{(2)} \leftarrow (e_{h_1}^{(2)}, \ldots, e_{h_n}^{(2)})$; $e_u^{(2)} \leftarrow p'_u + \sum_{i=1}^{\log n} l_{iu} x_i^2 + r_{iu} x_i^{-2} - w \cdot ab$

$e_g^{(2)} \leftarrow \sum_{i=1}^{\log n} l_{ig} x_i^2 + r_{ig} x_i^{-2} + p'_g$; $e_h^{(2)} \leftarrow \sum_{i=1}^{\log n} l_{ih} x_i^2 + r_{ih} x_i^{-2} + p'_h$

Return $(e_{\mathbf{g}}^{(2)}, e_{\mathbf{h}}^{(2)}, e_u^{(2)}, e_g^{(2)}, e_h^{(2)})$

Fig. 10. The function h for RngPf.

Proving this lemma would conclude the proof of Theorem 4.

□

Proof (Lemma 6).

In order to prove $g^{v^*} h^{\gamma^*} = V$ and $v^* \in [0, 2^n - 1]$, it suffices to show that $v_{\mathbf{g}} = v_{\mathbf{h}} = \mathbf{0}^n$, $v_u = 0$ and $v_g \in [0, 2^n - 1]$. Let us denote using $\tau|_c$ the partial transcript that is the prefix of τ just before the challenge c. For example

$$\tau|_{(y,z)} = \big((n, \mathbf{g}, \mathbf{h}, u, g, h), V, (A, S)\big)\ .$$

Since $\mathsf{h}([\tau])$ returns $(\mathbf{0}^n, \mathbf{0}^n, 0, 0, 0)$, we have that for $i = 1, 2$, $(e_{\mathbf{g}}^{(i)}, e_{\mathbf{h}}^{(i)}, e_g^{(i)}, e_h^{(i)}, e_u^{(i)}) = (\mathbf{0}^n, \mathbf{0}^n, 0, 0, 0)$.

Writing out the expression for $e_{\mathbf{g}}^{(1)}$ we get

$$v_{\mathbf{g}} z^2 + t_{1\mathbf{g}} x + t_{2\mathbf{g}} x^2 = \mathbf{0}^n\ .$$

Since $\tau \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{RngPf}}$, we have that $x \notin \mathsf{BadCh}(\tau|_x)$. Therefore, $\mathsf{SZ}(f_1(X), x)$ is `false` where f_1 is as defined in $\mathsf{CheckBad}(\tau', x)$. Since we have here that $f_1(x) = 0$, the polynomial $f_1(X)$ is the zero vector polynomial. In particular, its constant term $v_{\mathbf{g}} z^2 = \mathbf{0}^n$. Since $z \neq 0$ it follows that $v_{\mathbf{g}} = \mathbf{0}^n$. Similarly using $e_{\mathbf{h}}^{(1)} = \mathbf{0}^n$ and $e_u^{(1)} = 0$ we can show that $v_{\mathbf{h}} = \mathbf{0}^n$ and $v_u = 0$ respectively. Writing out the expression for $e_g^{(1)}$ we have $v_g z^2 + \delta(y,z) + t_{1g} x + t_{2g} x^2 - \hat{t} = 0$. Hence,

$$\hat{t} = v_g z^2 + \delta(y,z) + t_{1g} x + t_{2g} x^2\ . \tag{6}$$

Using $e_{\mathbf{g}}^{(2)} = \mathbf{0}^n$ we get for all $k \in \{0, \ldots, n-1\}$

$$p'_{g_{1+k}} + \sum_{i=1}^{\log n} (l_{ig_{1+k}} x_i^2 + r_{ig_{1+k}} x_i^{-2}) - a \cdot \prod_{i=1}^{\log n} x_i^{(-1)^{1-\mathsf{bit}(k,i,\log n)}} = 0\ . \tag{7}$$

Using $e_{\mathbf{h}}^{(2)} = \mathbf{0}^n$ we get for all $k \in \{0, \ldots, n-1\}$

$$p'_{h_{1+k}} + \sum_{i=1}^{\log n} (l_{ih_{1+k}} x_i^2 + r_{ih_{1+k}} x_i^{-2}) - by^{(-(k))} \cdot \prod_{i=1}^{\log n} x_i^{(-1)^{\mathsf{bit}(k,i,\log n)}} = 0 \,. \tag{8}$$

Using $e_u^{(2)} = 0$ we get that

$$p'_u + \sum_{i=1}^{\log n} (l_{iu} x_i^2 + r_{iu} x_i^{-2}) - w \cdot ab = 0 \,. \tag{9}$$

We shall next use the following lemma which essentially says that if all of $e_{\mathbf{g}}^{(2)}, e_{\mathbf{h}}^{(2)}, e_u^{(2)}, e_g^{(2)}, e_h^{(2)}$ are zero and $\tau \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{RngPf}}$, then $w \cdot \langle p'_{\mathbf{g}}, p'_{\mathbf{h}} \circ \mathbf{y}^n \rangle = p'_u$.

Lemma 7. *Let τ, as shown in (5), be an accepting transcript of* RngPf *such that $\tau \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{RngPf}}$. Let*

$$p'_{\mathbf{g}} = a_{\mathbf{g}} + x s_{\mathbf{g}} - z\mathbf{1}^n \,, p'_{\mathbf{h}} = a_{\mathbf{h}} + x s_{\mathbf{h}} + \mathbf{y}^{-n} \circ (z\mathbf{y}^n + z^2 \mathbf{2}^n) \,, p'_u = a_u + x s_u + w\hat{t} \,.$$

Suppose, the for all $k \in \{0, \ldots, n-1\}$

$$\left(\sum_{i=1}^{\log n} (l_{ig_{1+k}} x_i^2 + r_{ig_{1+k}} x_i^{-2}) + p'_{g_{1+k}} \right) - a \cdot \left(\prod_{i=1}^{\log n} x_i^{(-1)^{1-\mathsf{bit}(k,i,\log n)}} \right) = 0 \,,$$

$$\left(\sum_{i=1}^{\log n} (l_{ih_{1+k}} x_i^2 + r_{ih_{1+k}} x_i^{-2}) + p'_{h_{1+k}} \right) - by^{(-(k))} \cdot \left(\prod_{i=1}^{\log n} x_i^{(-1)^{\mathsf{bit}(k,i,\log n)}} \right) = 0 \,.$$

Also, $\left(\sum_{i=1}^{\log n} (l_{iu} x_i^2 + r_{iu} x_i^{-2}) \right) + p'_u - w \cdot ab = 0$. Then $w \cdot \langle p'_{\mathbf{g}}, p'_{\mathbf{h}} \circ \mathbf{y}^n \rangle = p'_u$.

The proof of this lemma is a generalization of the proof that we gave for the inner product argument for $n = 2$ in the technical overview.

Proof. We define a function Bad in Figure 11 that takes as input $x \in \mathbb{Z}_p^*$ and an index $m \in \{1, \ldots, \log n\}$. It returns `true` if and only if $x \in \mathsf{BadCh}(\tau|_{x_m})$. We shall then use Lemma 8, which is a purely algebraic lemma.

Lemma 8. *Let $n \in \mathbb{N}^+$ be a power of 2. Let $\{l_{i\mathbf{g}} \in \mathbb{Z}_p^n, l_{i\mathbf{h}} \in \mathbb{Z}_p^n, l_{iu} \in \mathbb{Z}_p, r_{i\mathbf{g}} \in \mathbb{Z}_p^n, r_{i\mathbf{h}} \in \mathbb{Z}_p^n, r_{iu} \in \mathbb{Z}_p\}_{i=1}^{\log n}$. Let $a, b, p'_{\mathbf{g}}, p'_{\mathbf{h}}, p'_u \in \mathbb{Z}_p$. Let*

$$\mathsf{params} = \left\{ \{l_{i\mathbf{g}}, l_{i\mathbf{h}}, l_{iu}, r_{i\mathbf{g}}, r_{i\mathbf{h}}, r_{iu}\}_{i=1}^{\log n}, p'_{\mathbf{g}}, p'_{\mathbf{h}}, p'_u \right\} \,.$$

Let $x_1, \ldots, x_{\log n} \in \mathbb{Z}_p^*$ such that $\mathsf{Bad}(\mathsf{params}, x_i, i) = \texttt{false}$ for $i = 1, \ldots, \log n$ where Bad *is defined in Figure 11. Suppose, the following equalities hold.*

1. *For all $k \in \{0, \ldots, n-1\}$*

$$\left(\sum_{i=1}^{\log n} (l_{ig_{1+k}} x_i^2 + r_{ig_{1+k}} x_i^{-2}) + p'_{g_{1+k}} \right) - a \cdot \left(\prod_{i=1}^{\log n} x_i^{(-1)^{1-\mathsf{bit}(k,i,\log n)}} \right) = 0 \,.$$

Procedure $\mathsf{Bad}(\mathsf{params}, x, m)$:

$//\mathsf{params} = \left\{\{l_{i\mathbf{g}}, l_{i\mathbf{h}}, l_{iu}, r_{i\mathbf{g}}, r_{i\mathbf{h}}, r_{iu}\}_{i=1}^{\log n}, p'_{\mathbf{g}}, p'_{\mathbf{h}}, p'_u\right\}$

For $j = 0, \ldots, n-1$ do

$f^{\mathbf{g}}_{m,j}(X) \leftarrow l_{mg_{1+j}} + X^2 + r_{mg_{1+j}} X^{-2} + p'_{g_{1+j}} + \sum_{i=1}^{m-1} (l_{ig_{1+j}} x_i^2 + r_{ig_{1+j}} x_i^{-2})$

$f^{\mathbf{h}}_{m,j}(X) \leftarrow l_{mh_{1+j}} X^2 + r_{mh_{1+j}} X^{-2} + p'_{h_{1+j}} + \sum_{i=1}^{m-1} (l_{ih_{1+j}} x_i^2 + r_{ih_{1+j}} x_i^{-2})$

$f^u_m(X) \leftarrow l_{mu} X^2 + r_{mu} X^{-2} + p'_u + \sum_{i=1}^{m-1} (l_{iu} x_i^2 + r_{iu} x_i^{-2})$

For $t = 1, \ldots, m-1$ do

For $j = 0, \ldots, n/2^t - 1$ do

$\mathsf{flag} \leftarrow \mathsf{flag} \vee \mathsf{SZ}(f^{\mathbf{g}}_{m,j}(X) \cdot x_t^2 - f^{\mathbf{g}}_{m,j+n/2^t}(X), x) \vee \mathsf{SZ}(f^{\mathbf{h}}_{m,j}(X) - f^{\mathbf{h}}_{m,j+n/2^t}(X) \cdot x_t^2, x)$

For $j = 0, \ldots, n/2^m - 1$ do

$\mathsf{flag} \leftarrow \mathsf{flag} \vee \mathsf{SZ}(f^{\mathbf{g}}_{m,j}(X) \cdot X^2 - f^{\mathbf{g}}_{m,j+n/2^m}(X), x) \vee \mathsf{SZ}(f^{\mathbf{h}}_{m,j}(X) - f^{\mathbf{h}}_{m,j+n/2^m}(X) \cdot X^2, x)$

$\mathsf{flag} \leftarrow \mathsf{flag} \vee \mathsf{SZ}\left(f^u_m(X) - w \cdot \sum_{j=0}^{n/2^m-1} f^{\mathbf{g}}_{m,j}(X) \cdot f^{\mathbf{h}}_{m,j}(X) \cdot y^j, x\right)$

Return flag

Fig. 11. The function Bad for Lemma 8.

2. *For all* $k \in \{0, \ldots, n-1\}$

$$\left(\sum_{i=1}^{\log n} (l_{ih_{1+k}} x_i^2 + r_{ih_{1+k}} x_i^{-2}) + p'_{h_{1+k}}\right) - by^{(-(k))} \cdot \left(\prod_{i=1}^{\log n} x_i^{(-1)^{\mathsf{bit}(k,i,\log n)}}\right) = 0 \ .$$

3.

$$\left(\sum_{i=1}^{\log n} (l_{iu} x_i^2 + r_{iu} x_i^{-2})\right) + p'_u - w \cdot ab = 0 \ .$$

Then

$$w \cdot \langle p'_{\mathbf{g}}, p'_{\mathbf{h}} \circ \mathbf{y}^n \rangle = p'_u \ .$$

Let $\mathsf{params} = \left\{\{l_{i\mathbf{g}}, l_{i\mathbf{h}}, l_{iu}, r_{i\mathbf{g}}, r_{i\mathbf{h}}, r_{iu}\}_{i=1}^{\log n}, p'_{\mathbf{g}}, p'_{\mathbf{h}}, p'_u\right\}$. Note that $\mathsf{Bad}(\mathsf{params}, x, j)$ returns `true` if and only if $x \in \mathsf{BadCh}(\tau|_{x_j})$. Therefore, we have that $x_1, \ldots, x_{\log n}$ in τ satisfy the condition for x_i's in Lemma 8. Moreover all the equalities required in Lemma 8 hold and $p'_{\mathbf{g}}, p'_{\mathbf{h}}, p'_u \in \mathbb{Z}_p$. So we using Lemma 8 we have that

$$w \cdot \langle p'_{\mathbf{g}}, p'_{\mathbf{h}} \circ \mathbf{y}^n \rangle = p'_u \ .$$

The proof of Lemma 8 is deferred to Section 5.4. □

Since τ is an accepting transcript of RngPf and $\tau \notin \mathcal{T}^{\mathsf{RngPf}}_{\mathsf{BadCh}}$ and (7) to (9) hold, using Lemma 7, we get $w\langle p'_{\mathbf{g}}, p'_{\mathbf{h}} \circ \mathbf{y}^n \rangle = p'_u$. Plugging in the values of $p'_{\mathbf{g}}, p'_{\mathbf{h}}, p'_u$ we get

$$w \cdot \langle a_{\mathbf{g}} + x s_{\mathbf{g}} - z\mathbf{1}^n, (a_{\mathbf{h}} + x s_{\mathbf{h}} + z\mathbf{1}^n) \circ \mathbf{y}^n + z^2 \mathbf{2}^n \rangle = a_u + x s_u + w\hat{t} \ .$$

Since $\tau \notin \mathcal{T}^{\mathsf{RngPf}}_{\mathsf{BadCh}}$, we have that $w \notin \mathsf{BadCh}(\tau|_w)$. Therefore, $\mathsf{SZ}(f(W), w)$ is `false` where f is as defined in $\mathsf{CheckBad}(\tau', w)$. Since we have here that $f(w) = 0$, the polynomial $f(W)$ must be the zero polynomial. In particular its W term must be zero, i.e.,

$$\langle a_{\mathbf{g}} + x s_{\mathbf{g}} - z\mathbf{1}^n, (a_{\mathbf{h}} + x s_{\mathbf{h}} + z\mathbf{1}^n) \circ \mathbf{y}^n + z^2 \mathbf{2}^n \rangle = \hat{t} \ .$$

Plugging in the value of $\hat{t}$ obtained in (6), we have that

$$(v_g z^2 + \delta(y,z) + t_{1g}x + t_{2g}x^2) - \langle a_{\mathbf{g}} + x s_{\mathbf{g}} - z\mathbf{1}^n, (a_{\mathbf{h}} + x s_{\mathbf{h}} + z\mathbf{1}^n) \circ \mathbf{y}^n + z^2\mathbf{2}^n \rangle = 0 \ .$$

Similarly, using $x \notin \mathsf{BadCh}(\tau|_x)$, we get

$$v_g z^2 + \delta(y,z) - \langle a_{\mathbf{g}} - z\mathbf{1}^n, (a_{\mathbf{h}} + z\mathbf{1}^n) \circ \mathbf{y}^n + z^2\mathbf{2}^n \rangle = 0 \ .$$

Plugging in the value of $\delta(y,z)$, rearranging and simplifying we get

$$z^2(v_g - \langle a_{\mathbf{g}}, \mathbf{2}^n \rangle) - z\langle a_{\mathbf{g}} - a_{\mathbf{h}} - \mathbf{1}^n, \mathbf{y}^n \rangle - \langle a_{\mathbf{g}} \circ a_{\mathbf{h}}, \mathbf{y}^n \rangle = 0 \ .$$

Using $(y,z) \notin \mathsf{BadCh}(\tau|_{(y,z)})$, we get that $v_g - \langle a_{\mathbf{g}}, \mathbf{2}^n \rangle = 0$, $a_{\mathbf{g}} - a_{\mathbf{h}} - \mathbf{1}^n = \mathbf{0}^n$, $a_{\mathbf{g}} \circ a_{\mathbf{h}} = \mathbf{0}^n$. Note that $a_{\mathbf{g}} - a_{\mathbf{h}} - \mathbf{1}^n = \mathbf{0}^n$ and $a_{\mathbf{g}} \circ a_{\mathbf{h}} = \mathbf{0}^n$ imply that $a_{\mathbf{g}} \in \{0,1\}^n$. Further $v_g - \langle a_{\mathbf{g}}, \mathbf{2}^n \rangle = 0$, i.e., $v_g = \langle a_{\mathbf{g}}, \mathbf{2}^n \rangle$. So, $v_g \in [0, 2^n - 1]$. Therefore, v^*, γ^* output by $\mathsf{e}([\tau])$ satisfy $V = g^{v^*} h^{\gamma^*}$ and $v^* \in [0, 2^n - 1]$. This concludes the proof of Lemma 6. □

Further for a prover $\mathcal{P}_{\mathsf{alg}}$ for $\mathsf{FS}^{\mathbf{RO}}[\mathsf{RngPf}]$, and the e we define in the proof of Theorem 4, we can upper bound $p_{\mathsf{fail},\mathsf{FS}}(\mathsf{FS}^{\mathbf{RO}}[\mathsf{RngPf}], \mathcal{P}_{\mathsf{alg}}, \mathsf{e}, R, \lambda)$ using techniques very similar to those used in the proof of Theorem 4. This is because we can prove that if the prover outputs an instance and an accepting proof and e fails to produce a valid witness, then we can compute a non-trivial discrete logarithm relation from the representation of the transcript and instance unless one of the challenges in the transcript are bad which we can show happens with small probability. Then using Theorem 3 we obtain a bound for the fs-ext-2 security of $\mathsf{FS}^{\mathbf{RO}}[\mathsf{RngPf}]$ similar to the one we obtained for fs-ext-1 security in Corollary 1.

TIGHTNESS OF THEOREM 4. We next argue that the factor $O(nq/(p-1))$ in Theorem 4 is tight. We first note that the protocol RngPf can be used for the following relation

$$R' = \left\{ (n \in \mathbb{N}, g, V \in \mathbb{G}, v \in \mathbb{Z}_p) : g^v = V \wedge v \in [0, 2^n - 1] \right\} , \tag{10}$$

by fixing γ to 0.

We shall construct a cheating prover $\mathcal{P}$ (that makes $O(q)$ queries to $\mathbf{O}_{\text{ext}}$) for the relation R' that outputs an instance $V = g^v$ such that $v \notin [0, 2^n - 1]$ but can still convince the RngPf verifier with probability $\Omega(nq/(p-1))$ if n divides $p-1$. In other words, we show that there exist n, p such that $\mathsf{Adv}^{\mathsf{srs}}_{\mathsf{RngPf}}(\mathcal{P}, \lambda) = \Omega(nq/(p-1))$. This would imply that for any $\lambda \in \mathbb{N}^+$, $\mathcal{D} = \mathsf{Acc}(.)$, $\mathsf{Adv}^{\mathsf{sr\text{-}wee}}_{\mathsf{RngPf},R}(\mathcal{P}_{\mathsf{alg}}, \mathcal{D}, \mathcal{E}, \lambda) = \Omega(nq/(p-1))$ for any extractor $\mathcal{E}$ – meaning that the bound in Theorem 4 is tight up to constant factors.

Theorem 5. *Let $\mathbb{G} = \{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}^+}$ be a family of groups of prime order $p = p(\lambda)$. Let $\mathsf{RngPf} = \mathsf{RngPf}[\mathbb{G}]$ be the interactive argument for the relation R' in (10) obtained by setting $\gamma = 0$ in the protocol defined in Figure 7. If n divides $p-1$, we can construct a non-uniform prover $\mathcal{P}$ making at most $q + \log n + 1$ queries to its oracle, such that for all $\lambda \in \mathbb{N}^+$, $\mathsf{Adv}^{\mathsf{srs}}_{\mathsf{RngPf}}(\mathcal{P}, \lambda) = \frac{(n-1)q}{p-1}$.*

Proof. In $\mathsf{SRS}_{\mathsf{RngPf}}$, on receiving $n, \mathbf{g}, \mathbf{h}, g, h, u$ as input, the first stage of $\mathcal{P}$ fixes $v = 2^{n+1} - 2$ and outputs $\mathsf{st}_{\mathcal{P}} = v$ and $V = g^v$. The second stage of the cheating prover $\mathcal{P}$ interacts with the game $\mathsf{SRS}_{\mathsf{RngPf}}$ as follows.

1. It initializes attempts $\leftarrow 0$.
2. If attempts $>= q$, it just aborts. Otherwise it increments attempts by 1.
3. It sets $\mathbf{a}_L = 2 \cdot \mathbf{1}^n, \mathbf{a}_R = \mathbf{1}^n$. It samples $\mathbf{s}_L, \mathbf{s}_R$ uniformly at random from $\mathbb{Z}_p^n$ and α, ρ uniformly at random from $\mathbb{Z}_p$. It computes $A = h^\alpha, S = h^\rho \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R}$ and queries $\mathbf{O}_{\text{ext}}$ with $(\varepsilon, (A, S))$ and receives y, z. In other words, it restores the state of the verifier to the initial state and sends A, S as the first message and receives y, z.
4. It checks if $\sum_{i=0}^{n-1} y^i = 0$. If the check succeeds, it moves to step 5. Otherwise it moves to step 2.
5. It now behaves like the honest prover $\mathsf{RngPf.P}$ till the end of the protocol. In particular, it does not attempt any more state-restorations.

First, we claim that if $\mathcal{P}$ reaches step 5, the game $\mathsf{SRS}_{\mathsf{RngPf}}$ outputs `true`. Since $\mathcal{P}$ behaves like the honest prover after it has sent A, S and received y, z it is easy to see that the $\mathsf{InPrd.V}$ shall return 1. We need to argue that the check $R = g^{\hat{t}} h^\tau$ succeeds. Since $\mathcal{P}$ behaves like an honest prover after receiving y, z, we have that

$$\hat{t} = t(x) = \langle l(x), r(x) \rangle = t_0 + t_1 x + t_2 x^2 \ .$$

This would give us

$$t_0 = \langle \mathbf{a}_L - z \cdot \mathbf{1}^n, \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n) + z^2 \cdot \mathbf{2}^n \rangle$$

Further, $\beta_x = \beta_1 x + \beta_2 x^2$, $R = V^{z^2} g^{\delta(y,z)} T_1^x T_2^{x^2} = g^{z^2 v + t_1 x + t_2 x^2 + \delta(y,z)} h^{\beta_1 x + \beta_2 x^2}$. Now since $\hat{t} = t_0 + t_1 x + t_2 x^2$ we have

$$\begin{aligned}(z^2 v + t_1 x + t_2 x^2 + \delta(y,z)) - \hat{t} = z^2 v + \delta(y,z) - t_0 = z^2 (v - \langle \mathbf{a}_L, \mathbf{2}^n \rangle) \\ - z \langle \mathbf{a}_L - \mathbf{a}_R - \mathbf{1}^n, \mathbf{y}^n \rangle - \langle \mathbf{a}_L \circ \mathbf{a}_R, \mathbf{y}^n \rangle \ .\end{aligned}$$

Since $\mathcal{P}$ had set $v = 2^{n+1} - 2, \mathbf{a}_L = 2 \cdot \mathbf{1}^n, \mathbf{a}_R = \mathbf{1}^n$ we have

$$(z^2 v + t_1 x + t_2 x^2 + \delta(y,z)) - \hat{t} = -2 \sum_{i=0}^{n-1} y^i = 0 \ .$$

Therefore

$$R = g^{z^2 v + t_1 x + t_2 x^2 + \delta(y,z)} h^{\beta_1 x + \beta_2 x^2} = g^{\hat{t}} h^{\beta_x} \ .$$

Hence, if $\mathcal{P}$ reaches step 5, the game $\mathsf{SRS}_{\mathsf{RngPf}}$ outputs `true`. We need to compute the probability that $\sum_{i=0}^{n-1} y^i = 0$ for a random y in $\mathbb{Z}_p^*$. First, we observe that

$$(y - 1) \sum_{i=0}^{n-1} y^i = 0 = y^n - 1 \ .$$

Now, if n divides $p - 1$, we claim that there are n distinct y's in $\mathbb{Z}_p^*$ that satisfy $y^n - 1 = 0$. Consider a generator g of $\mathbb{Z}_p^*$ (since p is a prime, the group $\mathbb{Z}_p^*$ is cyclic). Now g^j is a root of the equation $y^n - 1 = 0$ if $g^{jn} - 1 = 0$, i.e., if $p - 1$ divides jn. Since n divides $p - 1$, this condition is equivalent to $(p-1)/n$ divides j. So, g^j is a root of the equation $y^n - 1 = 0$ for $j = \{0, (p-1)/n, 2(p-1)/n, \ldots, (n-1)(p-1)/n\}$. In other words $y^n - 1 = 0$ has n distinct roots in $\mathbb{Z}_p^*$. So, the equation $\sum_{i=0}^{n-1} y^i = 0$ has $n - 1$ distinct roots because the factorization of a polynomial in a finite field is unique. Since y is picked uniformly at random, the probability that $\sum_{i=0}^{n-1} y^i = 0$ is $(n-1)/(p-1)$. Since $\mathcal{P}$ tries at most q different (A, S), the probability that it reaches step 5, is $(n-1)q/(p-1)$ – therefore $\mathsf{Adv}^{\mathsf{srs}}_{\mathsf{RngPf}}(\mathcal{P}, \lambda) = (n-1)q/(p-1)$. □

5.3 Online srs-wee Security for ACSPf

In this section, we introduce ACSPf and apply our framework to prove online srs-wee security. As shown in [BCC+16], any arithmetic circuit with n multiplication gates can be represented using a constraint system that has 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 $\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O$, with additional $Q \leqslant 2n$ linear constraints. The linear constraints can be represented as $\mathbf{a}_L \cdot \mathbf{W}_L + \mathbf{a}_R \cdot \mathbf{W}_R + \mathbf{a}_O \cdot \mathbf{W}_O = \mathbf{c}$, where $\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O \in \mathbb{Z}_p^{Q \times n}$.

We shall assume that $\mathsf{ACSPf} = \mathsf{ACSPf}[\mathbb{G}]$ is instantiated on an understood family of groups $\mathbb{G} = \{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}^+}$ of order $p = p(\lambda)$. The argument ACSPf is an argument of knowledge for the relation

$$R = \Big\{ \big((n, Q \in \mathbb{N}), (\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O \in \mathbb{Z}_p^{Q \times n}, \mathbf{c} \in \mathbb{Z}_p^Q), (\mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O \in \mathbb{Z}_p^n)\big) : \\ \mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O \wedge \mathbf{W}_L \cdot \mathbf{a}_L + \mathbf{W}_R \cdot \mathbf{a}_R + \mathbf{W}_O \cdot \mathbf{a}_O = \mathbf{c} \Big\} . \tag{11}$$

We note that in [BBB+18], an argument for a more generalized relation was given of which this is a special case. We can extend our proof for the more general relation. Here, for simplicity we only consider the above relation R that is enough for proving arithmetic circuit satisfiability.

DESCRIPTION OF ACSPf. The ACSPf.Setup procedure returns independent generators $\mathbf{g} \in \mathbb{G}^n, \mathbf{h} \in \mathbb{G}^n, g, h, u \in \mathbb{G}$ of the group $\mathbb{G}$. The instance for ACSPf is $\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O \in \mathbb{Z}_p^{Q \times n}, \mathbf{c} \in \mathbb{Z}_p^Q$ such that an honest prover knows a witness $(\mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O)$ that satisfies $\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O$ and $\mathbf{W}_L \cdot \mathbf{a}_L + \mathbf{W}_R \cdot \mathbf{a}_R + \mathbf{W}_O \cdot \mathbf{a}_O = \mathbf{c}$.

The prover and verifier for ACSPf is shown in Figure 12. The prover commits to $\mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O$ and proves to the verifier that these vectors satisfy the relation in (11). The prover and the verifier of ACSPf engage in InPrd in the final step to avoid the prover sending over vectors of length n.

We prove the following theorem that gives an upper bound on the online srs-wee security of ACSPf.

Theorem 6. *Let $\mathbb{G} = \{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}^+}$ be a family of groups of order $p = p(\lambda)$. Let $\mathsf{ACSPf} = \mathsf{ACSPf}[\mathbb{G}]$ be the interactive argument as defined in Figure 12, for the relation R in (11). We can construct an extractor $\mathcal{E}$ such that for any non-uniform algebraic prover $\mathcal{P}_{\mathsf{alg}}$ making at most $q = q(\lambda)$ queries to its oracle, there exists a non-uniform adversary $\mathcal{F}$ with the property that for any (computationally unbounded) distinguisher $\mathcal{D}$, for all $\lambda \in \mathbb{N}^+$*

$$\mathsf{Adv}^{\mathsf{sr\text{-}wee}}_{\mathsf{ACSPf},R}(\mathcal{P}_{\mathsf{alg}}, \mathcal{D}, \mathcal{E}, \lambda) \leqslant \frac{(14n+8)q}{p-1} + \mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{F}, \lambda) + \frac{1}{p} .$$

Moreover, the time complexity of the extractor $\mathcal{E}$ is $O(q \cdot n)$ and that of adversary $\mathcal{F}$ is $O(q \cdot n)$.

We can show that the bound in Theorem 6 is tight by constructing a cheating prover like we did in Theorem 5. Using Theorem 2, we get a corollary about fs-ext-1 security of $\mathsf{FS}^{\mathbf{RO}}[\mathsf{ACSPf}]$.

Corollary 2. *Let $\mathbb{G} = \{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}^+}$ be a family of groups of order $p = p(\lambda)$. Let $\mathsf{ACSPf} = \mathsf{ACSPf}[\mathbb{G}]$ be the interactive argument as defined in Figure 12, for the relation R in (11). Let $\mathsf{FS}^{\mathbf{RO}}[\mathsf{ACSPf}]$ be the non-interactive argument obtained by applying the Fiat-Shamir transform to ACSPf using a random oracle. We can construct an extractor $\mathcal{E}$ such that for any non-uniform algebraic prover $\mathcal{P}_{\mathsf{alg}}$ making at most $q = q(\lambda)$ queries to the random oracle there exists a non-uniform adversary $\mathcal{F}$ with the property that for all $\lambda \in \mathbb{N}^+$*

$$\mathsf{Adv}^{\mathsf{fs\text{-}ext\text{-}1}}_{\mathsf{FS}^{\mathbf{RO}}[\mathsf{ACSPf}],R}(\mathcal{P}_{\mathsf{alg}}, \mathcal{E}, \lambda) \leqslant \frac{(14n+9)q+1}{p-1} + \mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{F}, \lambda) + \frac{1}{p} .$$

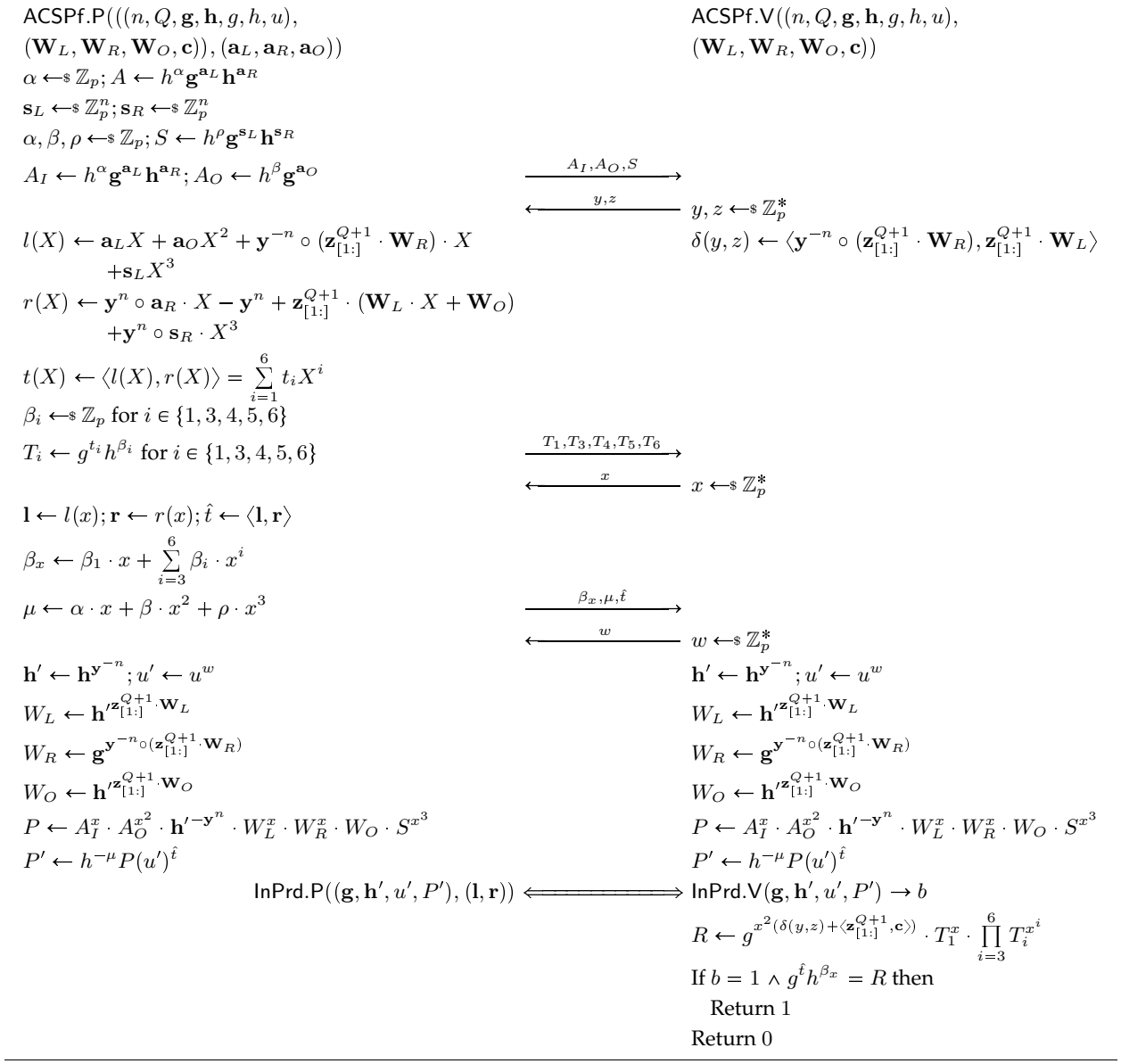

Fig. 12. Bulletproofs argument for arithmetic circuit satisfiability ACSPf.

Moreover, the time complexity of the extractor $\mathcal{E}$ is $O(q \cdot n)$ and that of adversary $\mathcal{F}$ is $O(q \cdot n)$.

Additionally, using techniques similar to those in the proof of Theorem 6, we can prove a similar bound for fs-ext-2 security of $\mathsf{FS}^{\mathbf{RO}}[\mathsf{ACSPf}]$.

Proof (Theorem 6). In order to prove this theorem, we invoke Theorem 1 by defining BadCh and e and showing that $\varepsilon \leqslant \frac{14n+8}{p-1}$ and there exists an adversary $\mathcal{F}$ such that $p_{\mathsf{fail}}(\mathsf{ACSPf}, \mathcal{P}_{\mathsf{alg}}, \mathsf{e}, R, \lambda) \leqslant \mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{F}) + \frac{1}{p}$.

DEFINING BadCh AND UPPER BOUNDING ε. To start off, we shall define $\mathsf{BadCh}(\tau')$ for all partial extended transcripts τ'. Let Ch be the set from which the challenge that comes right after τ' is sampled. We define a helper function $\mathsf{CheckBad}$ that takes as input a partial extended transcripts $[\tau']$ and a challenge $c \in \mathsf{Ch}$ and returns $\mathtt{true}$ if and only if $c \in \mathsf{BadCh}(\tau')$. For each verifier challenge in ACSPf, there is a definition of $\mathsf{CheckBad}$ in Figure 8. Every $\mathsf{CheckBad}$ function defines several bad conditions that depend on τ' – most of these bad conditions are checked using the predicate SZ (as defined before). One can safely ignore the details of the definitions of $\mathsf{CheckBad}$ functions for now – the rationale behind their definitions shall become apparent later on.

Next, we need to compute an upper bound ε on the size of $|\mathsf{BadCh}(\tau')|/|\mathsf{Ch}|$. To this end, we compute an upper bound on the maximum fraction of c's in Ch for which $\mathsf{CheckBad}(\tau', c)$ will return $\mathtt{true}$, for all the definitions of $\mathsf{CheckBad}$, using the Schwartz-Zippel Lemma.

The function $\mathsf{CheckBad}(\tau', (y, z))$ returns $\mathtt{true}$ if $\mathsf{SZ}(f(Y, Z), (y, z))$ is $\mathtt{true}$. The polynomial $f(Y, Z)$ is a polynomial of degree at most $n + 1$. So, the fraction of y, z for which $\mathsf{SZ}(f(Y, Z), (y, z))$ is $\mathtt{true}$ is at most $(n + 1)/(p - 1)$. So the the fraction of y, z in $\mathbb{Z}_p^*$ for which $\mathsf{CheckBad}(\tau', (y, z))$ returns $\mathtt{true}$ is at most $(n + 1)/(p - 1)$.

The function $\mathsf{CheckBad}(\tau', x)$ returns $\mathtt{true}$ if at least one of $\mathsf{SZ}(f_i(X), x)$ is $\mathtt{true}$ for $i \in [4]$. Since $f_1(X)$ and $f_2(X)$ are vector of n polynomials, each polynomial of degree 6, using the union bound the fraction of x's in $\mathbb{Z}_p^*$ for which $\mathsf{SZ}(f_1(X), x)$ or $\mathsf{SZ}(f_2(X), x)$ is $\mathtt{true}$ is at most $12n/(p - 1)$. The polynomial $f_3(X)$ is a polynomial of degree at most 6. The fraction of x's in $\mathbb{Z}_p^*$ for which $\mathsf{SZ}(f_3(X), x)$ is $\mathtt{true}$ is at most $6/(p - 1)$. The polynomial $f_4(X)$ is a polynomial of degree at most 4. The fraction of x's for which $\mathsf{SZ}(f_4(X), x)$ is $\mathtt{true}$ is at most $4/(p - 1)$. Using the union bound, the fraction of x's in $\mathbb{Z}_p^*$ for which $\mathsf{CheckBad}(\tau', x)$ returns $\mathtt{true}$ is at most $(12n + 10)/((p - 1))$.

The function $\mathsf{CheckBad}(\tau', w)$ returns $\mathtt{true}$ if $\mathsf{SZ}(f(W), w)$ is $\mathtt{true}$. The polynomial $f(W)$ is a polynomial of degree 1, hence using the Schwartz-Zippel Lemma the fraction of w's in $\mathbb{Z}_p^*$ for which $\mathsf{CheckBad}(\tau', w)$ returns $\mathtt{true}$ is at most $1/(p - 1)$.

The function $\mathsf{CheckBad}(\tau', x_m)$ for $m \in \{1, \ldots, \log n\}$ returns $\mathtt{true}$ if and only if SZ is $\mathtt{true}$ for any of the $\sum_{t=1}^{m-1} 2n/2^t$ polynomials of degree at most 4, $2n/2^m$ polynomials of degree at most 6 and one polynomial of degree at most 8. Using Schwartz Zippel Lemma and the union bound the fraction of x_m's in $\mathbb{Z}_p^*$ for which $\mathsf{CheckBad}(\tau', x_m)$ is $\mathtt{true}$ is at most

$$\frac{8}{p-1}\left(\sum_{t=1}^{m-1}\frac{n}{2^t}\right) + \frac{12n}{2^m(p-1)} + \frac{8}{p-1}\,.$$

This fraction is at most $(14n + 8)/(p - 1)$ for $m \in \{1, \ldots, \log n\}$. Therefore the fraction of c's in Ch for which $\mathsf{CheckBad}(\tau', c)$ will return $\mathtt{true}$ for any partial transcript τ' is upper bounded by $(14n + 8)/(p - 1)$, i.e., in the context of Theorem 1, $\varepsilon \leqslant \frac{14n+8}{p-1}$.

DEFINING e AND PROVING AN UPPER BOUND ON $p_{\mathsf{fail}}(\mathsf{ACSPf}, \mathcal{P}_{\mathsf{alg}}, \mathsf{e}, R, \lambda)$. The function e simply outputs $(a_{I\mathbf{g}}, a_{I\mathbf{h}}, a_{O\mathbf{g}})$ and outputs them. It follows from the description of e that it runs in time

Procedure $\mathsf{CheckBad}(\lceil\tau'\rceil, (y,z))$:
$//\lceil\tau'\rceil = \left((n,Q,\mathbf{g},\mathbf{h},u,g,h),(\mathbf{W}_L,\mathbf{W}_R,\mathbf{W}_O,\mathbf{c});([A_I],[A_O],[S])\right)$
$f(Y,Z) \leftarrow \langle \mathbf{Z}_{[1:]}^{Q+1}, \mathbf{c} - \mathbf{W}_L \cdot a_{I\mathbf{g}} - \mathbf{W}_R \cdot a_{I\mathbf{h}} - \mathbf{W}_O \cdot a_{O\mathbf{g}} \rangle - \langle a_{I\mathbf{g}} \circ a_{I\mathbf{h}} - a_{O\mathbf{g}}, \mathbf{Y}^n \rangle$
Return $\mathsf{SZ}(f(Y,Z),(y,z))$

Procedure $\mathsf{CheckBad}(\lceil\tau'\rceil, x)$:
$//\lceil\tau'\rceil = \left((n,Q,\mathbf{g},\mathbf{h},u,g,h),(\mathbf{W}_L,\mathbf{W}_R,\mathbf{W}_O,\mathbf{c});([A_I],[A_O],[S]),(y,z),([T_1],[T_3],[T_4],[T_5],[T_6])\right)$
$f_1(X) \leftarrow t_{1\mathbf{g}}X + \sum_{i=3}^{6} t_{i\mathbf{g}}X^i$; $f_2(X) \leftarrow t_{1\mathbf{h}}X + \sum_{i=3}^{6} t_{i\mathbf{h}}X^i$; $f_3(X) \leftarrow t_{1u}X + \sum_{i=3}^{6} t_{iu}X^i$
$l(X) \leftarrow a_{I\mathbf{g}} \cdot X + a_{O\mathbf{g}} \cdot X^2 + \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_R) \cdot X + s_{\mathbf{h}} \cdot X^3$; $r(X) \leftarrow \mathbf{y}^n \circ a_{I\mathbf{h}} \cdot X - \mathbf{y}^n + \mathbf{z}_{[1:]}^{Q+1} \cdot (\mathbf{W}_L \cdot X + \mathbf{W}_O) + \mathbf{y}^n \circ s_{\mathbf{h}}$
$\delta(y,z) \leftarrow \langle \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_R), \mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_L \rangle$; $f_4(X) \leftarrow X^2(\delta(y,z) + \langle \mathbf{z}_{[1:]}^{Q+1}, \mathbf{c} \rangle) + t_{1g}X + \sum_{i=1}^{3} t_{ig}X^i - \langle l(X), r(X) \rangle$
Return $\mathsf{SZ}(f_1(X),x) \vee \mathsf{SZ}(f_2(X),x) \vee \mathsf{SZ}(f_3(X),x) \vee \mathsf{SZ}(f_4(X),x)$

Procedure $\mathsf{CheckBad}(\lceil\tau'\rceil, w)$:
$//\lceil\tau'\rceil = \left((n,Q,\mathbf{g},\mathbf{h},u,g,h),(\mathbf{W}_L,\mathbf{W}_R,\mathbf{W}_O,\mathbf{c});([A_I],[A_O],[S]),(y,z),([T_1],[T_3],[T_4],[T_5],[T_6]),x,(\beta_x,\mu,\hat{t})\right)$
$\mathbf{l} \leftarrow a_{I\mathbf{g}} \cdot x + a_{O\mathbf{g}} \cdot x^2 + \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_R) \cdot x + s_{\mathbf{h}} \cdot x^3$
$\mathbf{r} \leftarrow \mathbf{y}^n \circ a_{I\mathbf{h}} \cdot x - \mathbf{y}^n + \mathbf{z}_{[1:]}^{Q+1} \cdot (\mathbf{W}_L \cdot x + \mathbf{W}_O) + \mathbf{y}^n \circ s_{\mathbf{h}}$; $f(W) \leftarrow W\hat{t} - W\langle \mathbf{l}, \mathbf{r} \rangle$
Return $\mathsf{SZ}(f(W),w)$

Procedure $\mathsf{CheckBad}(\lceil\tau'\rceil, x_m)$:
$//\lceil\tau'\rceil = \left((n,Q,\mathbf{g},\mathbf{h},u,g,h),(\mathbf{W}_L,\mathbf{W}_R,\mathbf{W}_O,\mathbf{c});([A_I],[A_O],[S]),(y,z),([T_1],[T_3],[T_4],[T_5],[T_6]),x,(\beta_x,\mu,\hat{t}),w,\right.$
$\left.([L_1],[R_1]),x_1,\ldots,([L_m],[R_m])\right)$
$p'_{\mathbf{g}} \leftarrow a_{I\mathbf{g}} \cdot x + a_{O\mathbf{g}} \cdot x^2 + \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_R) \cdot x + s_{\mathbf{g}} \cdot x^3$; $p'_u \leftarrow a_{Iu} \cdot x + a_{Iu} \cdot x^2 + s_u \cdot x^3 + w\hat{t}$
$p'_{\mathbf{h}} \leftarrow a_{I\mathbf{h}} \cdot x + a_{O\mathbf{h}} \cdot x^2 - \mathbf{1}^n + \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_L) \cdot x + \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_O) + s_{\mathbf{g}} \cdot x^3$
For $j = 0,\ldots,n-1$ do
  $f_{m,j}^{\mathbf{g}}(X) \leftarrow l_{mg_{1+j}} + X^2 + r_{mg_{1+j}}X^{-2} + p'_{g_{1+j}} + \sum_{i=1}^{m-1}(l_{ig_{1+j}}x_i^2 + r_{ig_{1+j}}x_i^{-2})$
  $f_{m,j}^{\mathbf{h}}(X) \leftarrow l_{mh_{1+j}}X^2 + r_{mh_{1+j}}X^{-2} + p'_{h_{1+j}} + \sum_{i=1}^{m-1}(l_{ih_{1+j}}x_i^2 + r_{ih_{1+j}}x_i^{-2})$
$f_m^u(X) \leftarrow l_{mu}X^2 + r_{mu}X^{-2} + p'_u + \sum_{i=1}^{m-1}(l_{iu}x_i^2 + r_{iu}x_i^{-2})$
flag $\leftarrow$ `false`
For $t = 1,\ldots,m-1$ do
  For $j = 0,\ldots,n/2^t - 1$ do
    flag $\leftarrow$ flag $\vee$ $\mathsf{SZ}(f_{m,j}^{\mathbf{g}}(X) \cdot x_t^2 - f_{m,j+n/2^t}^{\mathbf{g}}(X), x_m) \vee \mathsf{SZ}(f_{m,j}^{\mathbf{h}}(X) - f_{m,j+n/2^t}^{\mathbf{h}}(X) \cdot x_t^2, x_m)$
For $j = 0,\ldots,n/2^m - 1$ do
  flag $\leftarrow$ flag $\vee$ $\mathsf{SZ}(f_{m,j}^{\mathbf{g}}(X) \cdot X^2 - f_{m,j+n/2^m}^{\mathbf{g}}(X), x_m) \vee \mathsf{SZ}(f_{m,j}^{\mathbf{h}}(X) - f_{m,j+n/2^m}^{\mathbf{h}}(X) \cdot X^2, x_m)$
flag $\leftarrow$ flag $\vee$ $\mathsf{SZ}\left(f_m^u(X) - w \cdot \sum_{j=0}^{n/2^m-1} f_{m,j}^{\mathbf{g}}(X) \cdot f_{m,j}^{\mathbf{h}}(X) \cdot y^j, x_m\right)$
Return flag

Fig. 13. The function CheckBad function for the ACSPf.

Procedure $\mathsf{e}([\tau])$:

$//[\tau] = \big((n, Q, \mathbf{g}, \mathbf{h}, u, g, h), (\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c}); ([A], [S]), (y, z), ([T_1], [T_3], [T_4], [T_5], [T_6]), x, (\beta_x, \mu, \hat{t}), w, ([L_1], [R_1]), x_1, \ldots, ([L_{\log n}], [R_{\log n}]), x_{\log n}, (a, b)\big)$

Return $(a_{I\mathbf{g}}, a_{I\mathbf{h}}, a_{O_\mathbf{g}})$

Fig. 14. The function e for ACSPf.

Procedure $\mathsf{h}([\tau])$:

$//[\tau] = \big((n, Q, \mathbf{g}, \mathbf{h}, u, g, h), (\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c}); ([A], [S]), (y, z), ([T_1], [T_3], [T_4], [T_5], [T_6]), x, (\beta_x, \mu, \hat{t}), w, ([L_1], [R_1]), x_1, \ldots, ([L_{\log n}], [R_{\log n}]), x_{\log n}, (a, b)\big)$

$\delta(y,z) \leftarrow \langle \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_R), \mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_L \rangle$; $e_\mathbf{g}^{(1)} \leftarrow t_{1\mathbf{g}} x + \sum_{i=3}^{6} t_{i\mathbf{g}} x^i$; $e_\mathbf{h}^{(1)} \leftarrow t_{1\mathbf{h}} x + \sum_{i=3}^{6} t_{i\mathbf{h}} x^i$; $e_u^{(1)} \leftarrow t_{1u} x + \sum_{i=3}^{6} t_{iu} x^i$

$e_g^{(1)} \leftarrow x^2(\delta(y,z) + \langle \mathbf{z}_{[1:]}^{Q+1}, \mathbf{c} \rangle) + t_{1g} x + \sum_{i=3}^{6} t_{ig} x^i - \hat{t}$; $e_h^{(1)} \leftarrow t_{1h} x + \sum_{i=3}^{6} t_{ih} x^i - \beta_x$

If $(e_\mathbf{g}^{(1)}, e_\mathbf{h}^{(1)}, e_u^{(1)}, e_g^{(1)}, e_h^{(1)}) \neq (\mathbf{0}^n, \mathbf{0}^n, 0, 0, 0)$ then

  Return $(e_\mathbf{g}^{(1)}, e_\mathbf{h}^{(1)}, e_u^{(1)}, e_g^{(1)}, e_h^{(1)})$

$p'_\mathbf{g} \leftarrow a_{I\mathbf{g}} \cdot x + a_{O\mathbf{g}} \cdot x^2 + \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_R) \cdot x + s_\mathbf{g} \cdot x^3$

$p'_\mathbf{h} \leftarrow a_{I\mathbf{h}} \cdot x + a_{O\mathbf{h}} \cdot x^2 - \mathbf{1}^n + \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_L) \cdot x + \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_O) + s_\mathbf{g} \cdot x^3$

$p'_g \leftarrow a_{Ig} \cdot x + a_{Ig} \cdot x^2 + s_g \cdot x^3$; $p'_h \leftarrow a_{Ih} \cdot x + a_{Ih} \cdot x^2 + s_h \cdot x^3 - \mu$; $p'_u \leftarrow a_{Iu} \cdot x + a_{Iu} \cdot x^2 + s_u \cdot x^3 + w\hat{t}$; $p'_V \leftarrow a_V + x s_V$

For $k = 0$ to $n - 1$ do

  $e_{g_{k+1}}^{(2)} \leftarrow \Big(\sum_{i=1}^{\log n} (l_{ig_{1+k}} x_i^2 + r_{ig_{1+k}} x_i^{-2}) + p'_{g_{1+k}} \Big) - a \cdot \Big(\prod_{i=1}^{\log n} x_i^{(-1)^{1-\mathsf{bit}(k,i,\log n)}} \Big)$

  $e_{h_{k+1}}^{(2)} \leftarrow \Big(\sum_{i=1}^{\log n} (l_{ih_{1+k}} x_i^2 + r_{ih_{1+k}} x_i^{-2}) + p'_{h_{1+k}} \Big) - b y^{(-(k))} \cdot \Big(\prod_{i=1}^{\log n} x_i^{(-1)^{\mathsf{bit}(k,i,r)}} \Big)$

$e_\mathbf{g}^{(2)} \leftarrow (e_{g_1}^{(2)}, \ldots, e_{g_n}^{(2)})$; $e_\mathbf{h}^{(2)} \leftarrow (e_{h_1}^{(2)}, \ldots, e_{h_n}^{(2)})$

$e_u^{(2)} \leftarrow \Big(\sum_{i=1}^{\log n} (l_{iu} x_i^2 + r_{iu} x_i^{-2}) + p'_u \Big) - w \cdot ab$; $e_g^{(2)} \leftarrow \Big(\sum_{i=1}^{\log n} l_{ig} x_i^2 + r_{ig} x_i^{-2} \Big) + p'_g$; $e_h^{(2)} \leftarrow \Big(\sum_{i=1}^{\log n} l_{ih} x_i^2 + r_{ih} x_i^{-2} \Big) + p'_h$

Return $(e_\mathbf{g}^{(2)}, e_\mathbf{h}^{(2)}, e_u^{(2)}, e_g^{(2)}, e_h^{(2)})$

Fig. 15. The function h for ACSPf.

$O(n)$. Note that ACSPf.V runs in time $O(n)$. Therefore, using Theorem 1, the time complexity of $\mathcal{E}$ is $O(q \cdot n)$.

In order to complete our proof we need to upper bound $p_{\mathsf{fail}}(\mathsf{ACSPf}, \mathcal{P}_{\mathsf{alg}}, \mathsf{e}, R, \lambda)$. To do so we shall construct an adversary $\mathcal{H}$ (that runs $\mathcal{P}_{\mathsf{alg}}$) against that takes as input independent generators $\mathbf{g}, \mathbf{h}, g, h, u$ of the group $\mathbb{G}$ and finds a non-trivial discrete logarithm relation between them, i.e., computes $(e_\mathbf{g}, e_\mathbf{h}, e_g, e_h, e_u) \neq (\mathbf{0}^n, \mathbf{0}^n, 0, 0, 0)$ such that $\mathbf{g}^{e_\mathbf{g}} \mathbf{h}^{e_\mathbf{h}} g^{e_g} h^{e_h} u^{e_u} = 1$. Then we shall invoke Lemma 2 to transform $\mathcal{H}$ into an $\mathcal{F}$ against the discrete logarithm problem.

The adversary $\mathcal{H}$ has inputs $\mathbf{g}, \mathbf{h}, g, h, u$, it chooses $Q \leqslant 2n$ and runs $\mathcal{P}_{\mathsf{alg}}$ on public parameters $n, Q, \mathbf{g}, \mathbf{h}, g, h, u$ and simulates the game $\mathsf{SRS}_{\mathsf{ACSPf}}$ to it. If $\mathcal{P}_{\mathsf{alg}}$ manages to produce an accepting transcript τ, $\mathcal{H}$ calls a helper function h on input $[\tau]$ and outputs whatever h outputs.

DEFINING h. The function h is defined in Figure 15. It follows from the description of h that it runs time at most $O(n)$. The running time of $\mathcal{H}$ consists of the time required to answers q queries, run ACSPf.V in at most q paths in the execution tree and the time required to run h. Hence its time complexity is $O(q \cdot n)$. Using Lemma 2, time complexity of $\mathcal{F}$ is $O(q \cdot n)$.

We shall next discuss the rationale behind the definition of h. Let τ be a transcript of ACSPf as shown below.

$$\begin{aligned}\tau = \big(& (n, Q, \mathbf{g}, \mathbf{h}, u, g, h), (\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c}); (A_I, A_O, S), (y, z), (T_1, T_3, T_4, T_5, T_6), x, (\beta_x, \mu, \hat{t}), w, \\ & (L_1, R_1), x_1, (L_2, R_2), x_2, \ldots, (L_{\log n}, R_{\log n}), x_{\log n}, (a, b)\big) \ .\end{aligned} \tag{12}$$

The following equality must hold if τ is an accepting transcript.

$$g^{\hat{t}} h^{\beta_x} = g^{x^2(\delta(y,z) + \langle \mathbf{z}_{[1:]}^{Q+1}, \mathbf{c} \rangle)} T_1^x \cdot \prod_{i=3}^{6} T_i^{x^i} \ .$$

Writing out T_1, T_3, T_4, T_5, T_6 in terms of their representations and rearranging we shall get that

$$\mathbf{g}^{e_{\mathbf{g}}^{(1)}} \mathbf{h}^{e_{\mathbf{h}}^{(1)}} g^{e_g^{(1)}} h^{e_h^{(1)}} u^{e_u^{(1)}} = 1 \ ,$$

where $e_{\mathbf{g}}^{(1)}, e_{\mathbf{h}}^{(1)}, e_g^{(1)}, e_h^{(1)}, e_u^{(1)}$ are as defined in h. Again since τ is an accepting transcript the inner product verifier must have returned 1 and hence the following equality must hold.

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

Now we can write the left hand side of the above equality as

$$\begin{aligned}& \left(\prod_{i=1}^{\log n} L_i^{x_i^2} \right) h^{-\mu} A_I^x \cdot A_O^{x^2} \cdot \mathbf{h}^{\mathbf{1}^n} \cdot (\mathbf{h}^{\mathbf{y}^{-n}})^{\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_L{}^x} \cdot \mathbf{g}^{\mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_R)^x} \\ & \cdot (\mathbf{h}^{\mathbf{y}^{-n}})^{\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_O} \cdot S^{x^3} \cdot (u^w)^{\hat{t}} \cdot \left(\prod_{i=1}^{\log n} R_i^{x_i^{-2}} \right) \ .\end{aligned}$$

Let the function $\mathsf{bit}(k, i, t)$ return the bit k_i where $(k_1, \ldots, k_t)$ is the t-bit representation of k. Then we can write

$$\mathbf{g}^{(\log n)} = \prod_{k=0}^{n-1} g_{1+k}^{\prod_{i=1}^{\log n} x_i^{(-1)^{1-\mathsf{bit}(k,i,\log n)}}} \ ,$$

and

$$\mathbf{h}^{(\log n)} = \prod_{k=0}^{n-1} h_{1+k}^{y^{(-1+k)} \prod_{i=1}^{\log n} x_i^{(-1)^{\mathsf{bit}(k,i,\log n)}}} \ .$$

Plugging these into the inequality and rearranging we shall get that

$$\mathbf{g}^{e_{\mathbf{g}}^{(2)}} \mathbf{h}^{e_{\mathbf{h}}^{(2)}} g^{e_g^{(2)}} h^{e_h^{(2)}} u^{e_u^{(2)}} = 1 \ ,$$

where $e_{\mathbf{g}}^{(2)}, e_{\mathbf{h}}^{(2)}, e_g^{(2)}, e_h^{(2)}, e_u^{(2)}$ are as defined in h.

Therefore, h always returns a valid discrete logarithm relation when it gets an accepting transcript as input.

RELATING h, e. In order to complete the proof of Theorem 6, in the following lemma we show that – on an accepting transcript τ such that $\tau \notin \mathcal{T}^{\mathsf{ACSPf}}_{\mathsf{BadCh}}$ if $\mathsf{h}([\tau])$ returns a trivial discrete logarithm relation, then $\mathsf{e}([\tau])$ returns a valid witness.

Lemma 9. *Let τ, as shown in (12), be an accepting transcript of ACSPf such that $\tau \notin \mathcal{T}^{\mathsf{ACSPf}}_{\mathsf{BadCh}}$. If $\mathsf{h}([\tau])$ returns $(\mathbf{0}^n, \mathbf{0}^n, 0, 0, 0)$ then $\mathsf{e}([\tau])$ returns $(\mathbf{a}^*_L, \mathbf{a}^*_R, \mathbf{a}^*_O)$ such that*

$$\mathbf{a}^*_L \circ \mathbf{a}^*_R = \mathbf{a}^*_O \text{ and } \mathbf{W}_L \cdot \mathbf{a}^*_L + \mathbf{W}_R \cdot \mathbf{a}^*_R + \mathbf{W}_O \cdot \mathbf{a}^*_O = \mathbf{c} \ .$$

Taking the contrapositive, we have that whenever $\mathsf{e}([\tau])$ fails to extract a valid witness for an accepting transcript $\tau \notin \mathcal{T}^{\mathsf{ACSPf}}_{\mathsf{BadCh}}$, $\mathsf{h}([\tau])$ outputs a non-trivial discrete logarithm relation, i.e., $\mathcal{H}$ succeeds. So we have that

$$p_{\mathsf{fail}}(\mathsf{ACSPf}, \mathcal{P}_{\mathsf{alg}}, \mathsf{e}, R, \lambda) \leqslant \mathsf{Adv}^{\mathsf{dl\text{-}rel}}_{\mathbb{G}, 2n+3}(\mathcal{H})$$

Using Lemma 2 we would have that there exists an adversary $\mathcal{F}$ such that

$$p_{\mathsf{fail}}(\mathsf{ACSPf}, \mathcal{P}_{\mathsf{alg}}, \mathsf{e}, R, \lambda) \leqslant \mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{F}) + \frac{1}{p} \ .$$

Moreover, $\mathcal{F}$ is nearly as efficient as $\mathcal{H}$.

□

We next prove Lemma 9.

Proof (Lemma 9). For simplicity let us represent using $\tau|_c$ the prefix of τ just before the challenge c. For example

$$\tau|_{(y,z)} = \big((n, Q, \mathbf{g}, \mathbf{h}, u, g, h), (\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c}), (A_I, A_O, S)\big) \ .$$

Since $\mathsf{h}([\tau])$ returns $(\mathbf{0}^n, \mathbf{0}^n, 0, 0, 0)$, we have that for $i = 1, 2$

$$(e^{(i)}_{\mathbf{g}}, e^{(i)}_{\mathbf{h}}, e^{(i)}_g, e^{(i)}_h, e^{(i)}_u) = (\mathbf{0}^n, \mathbf{0}^n, 0, 0, 0) \ .$$

Since $e^{(1)}_g = 0$ we have that $x^2(\delta(y,z) + \langle \mathbf{z}^{Q+1}_{[1:]}, \mathbf{c}\rangle) + t_{1g}x + \sum_{i=3}^{6} t_{ig}x^i - \hat{t} = 0$. Hence

$$\hat{t} = x^2(\delta(y,z) + \langle \mathbf{z}^{Q+1}_{[1:]}, \mathbf{c}\rangle) + t_{1g}x + \sum_{i=3}^{6} t_{ig}x^i \ . \tag{13}$$

We define $p'_{\mathbf{g}}, p'_{\mathbf{h}}, p'_u$ as defined in h, i.e.,

$$\begin{aligned}
p'_{\mathbf{g}} &= a_{I\mathbf{g}} \cdot x + a_{O\mathbf{g}} \cdot x^2 + \mathbf{y}^{-n} \circ (\mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_R) \cdot x + s_{\mathbf{g}} \cdot x^3 \ , \\
p'_{\mathbf{h}} &= a_{I\mathbf{h}} \cdot x + a_{O\mathbf{h}} \cdot x^2 - \mathbf{1}^n + \mathbf{y}^{-n} \circ (\mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_L) \cdot x + \mathbf{y}^{-n} \circ (\mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_O) + s_{\mathbf{g}} \cdot x^3 \ , \\
p'_u &= a_{Iu} \cdot x + a_{Iu} \cdot x^2 + s_u \cdot x^3 + w\hat{t} \ .
\end{aligned}$$

Since $e^{(2)}_{\mathbf{g}} = \mathbf{0}^n$, we have that for all $k \in \{0, \ldots, n-1\}$

$$\left(\sum_{i=1}^{\log n} (l_{ig_{1+k}} x_i^2 + r_{ig_{1+k}} x_i^{-2}) + p'_{g_{1+k}} \right) - a \cdot \left(\prod_{i=1}^{\log n} x_i^{(-1)^{1-\mathsf{bit}(k,i,\log n)}} \right) = 0 \ . \tag{14}$$

We also have $e_{\mathbf{h}}^{(2)} = \mathbf{0}^n$, i.e., for all $k \in \{0, \ldots, n-1\}$

$$\left(\sum_{i=1}^{\log n} (l_{ih_{1+k}} x_i^2 + r_{ih_{1+k}} x_i^{-2}) + p'_{h_{1+k}} \right) - by^{(-(k))} \cdot \left(\prod_{i=1}^{\log n} x_i^{(-1)^{\mathsf{bit}(k,i,\log n)}} \right) = 0 \,. \tag{15}$$

From $e_u^{(2)} = 0$ we have that

$$\left(\sum_{i=1}^{\log n} (l_{iu} x_i^2 + r_{iu} x_i^{-2}) \right) + p'_u - w \cdot ab = 0 \,. \tag{16}$$

We shall next use the following lemma which essentially says that if none of $e_{\mathbf{g}}^{(2)}, e_{\mathbf{h}}^{(2)}, e_u^{(2)}, e_g^{(2)}, e_h^{(2)}$ are non-zero and $\tau \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{ACSPf}}$, then $w \cdot \langle p'_{\mathbf{g}}, p'_{\mathbf{h}} \circ \mathbf{y}^n \rangle = p'_u$. It is very similar to Lemma 7 that we encountered in the analysis of RngPf. This similarity is due to both ACSPf and RngPf use the inner-product argument.

The equalities in the statement of this lemma hold if the inner-product argument verifier accepts and the discrete logarithm problem is hard in group $\mathbb{G}$. The lemma shows that if none of the challenges in the inner-product argument were bad, then the inner-product of the vectors $p'_{\mathbf{g}}$ and $p'_{\mathbf{h}} \circ \mathbf{y}^n$ is p'_u/w. This is a generalization of the proof that we saw in the technical overview where we analysed the inner-product argument for $n = 2$.

Lemma 10. *Let τ, as shown in (12), be an accepting transcript of* ACSPf *such that $\tau \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{ACSPf}}$. Let*

$$\begin{aligned}
p'_{\mathbf{g}} &= a_{I\mathbf{g}} \cdot x + a_{O\mathbf{g}} \cdot x^2 + \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_R) \cdot x + s_{\mathbf{g}} \cdot x^3 \,, \\
p'_{\mathbf{h}} &= a_{I\mathbf{h}} \cdot x + a_{O\mathbf{h}} \cdot x^2 - \mathbf{1}^n + \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_L) \cdot x + \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_O) + s_{\mathbf{g}} \cdot x^3 \,, \\
p'_u &= a_{Iu} \cdot x + a_{Iu} \cdot x^2 + s_u \cdot x^3 + w\hat{t} \,.
\end{aligned}$$

Suppose, the for all $k \in \{0, \ldots, n-1\}$

$$\left(\sum_{i=1}^{\log n} (l_{ig_{1+k}} x_i^2 + r_{ig_{1+k}} x_i^{-2}) + p'_{g_{1+k}} \right) - a \cdot \left(\prod_{i=1}^{\log n} x_i^{(-1)^{1-\mathsf{bit}(k,i,\log n)}} \right) = 0 \,,$$

$$\left(\sum_{i=1}^{\log n} (l_{ih_{1+k}} x_i^2 + r_{ih_{1+k}} x_i^{-2}) + p'_{h_{1+k}} \right) - by^{(-(k))} \cdot \left(\prod_{i=1}^{\log n} x_i^{(-1)^{\mathsf{bit}(k,i,\log n)}} \right) = 0 \,.$$

Additionally,

$$\left(\sum_{i=1}^{\log n} (l_{iu} x_i^2 + r_{iu} x_i^{-2}) \right) + p'_u - w \cdot ab = 0 \,. \tag{17}$$

Then

$$w \cdot \langle p'_{\mathbf{g}}, p'_{\mathbf{h}} \circ \mathbf{y}^n \rangle = p'_u \,.$$

Proof (Lemma 10). We shall invoke Lemma 8 to prove this lemma. Let

$$\mathsf{params} = \left\{ \{l_{i\mathbf{g}}, l_{i\mathbf{h}}, l_{iu}, r_{i\mathbf{g}}, r_{i\mathbf{h}}, r_{iu}\}_{i=1}^{\log n}, p'_{\mathbf{g}}, p'_{\mathbf{h}}, p'_u \right\} \,.$$

Consider the function Bad defined in Figure 11. Note that since $\mathsf{Bad}(\mathsf{params}, x, j)$ returns `true` if and only if $x \in \mathsf{BadCh}(\tau|_{x_j})$, $x_1, \ldots, x_{\log n}$ in τ satisfy the condition for x_i's in Lemma 8. Moreover all the equalities required in Lemma 8 hold and $p'_{\mathbf{g}}, p'_{\mathbf{h}}, p'_u \in \mathbb{Z}_p$. So we using Lemma 8 we have that

$$w \cdot \langle p'_{\mathbf{g}}, p'_{\mathbf{h}} \circ \mathbf{y}^n \rangle = p'_u \,.$$

□

Since τ is an accepting transcript of ACSPf and $\tau \notin \mathcal{T}^{\mathsf{ACSPf}}_{\mathsf{BadCh}}$ and (14) to (16) hold, using Lemma 10, we get

$$w\langle p'_{\mathbf{g}}, p'_{\mathbf{h}} \circ \mathbf{y}^n \rangle = p'_u \ .$$

Plugging in the values of $p'_{\mathbf{g}}, p'_{\mathbf{h}}, p'_u$ we get

$$w \cdot \Big\langle \big(a_{I\mathbf{g}} x + a_{O\mathbf{g}} x^2 + \mathbf{y}^{-n} \circ (\mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_R) x + s_{\mathbf{g}} x^3\big), \mathbf{y}^n \circ \big(a_{I\mathbf{h}} x + a_{O\mathbf{h}} x^2 - \mathbf{1}^n + \mathbf{y}^{-n} \circ (\mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_L) \cdot x + \mathbf{y}^{-n} \circ (\mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_O) + s_{\mathbf{g}} x^3\big) \Big\rangle = a_{Iu} x + a_{Iu} x^2 + s_u x^3 + w\hat{t} \ .$$

Since $\tau \notin \mathcal{T}^{\mathsf{ACSPf}}_{\mathsf{BadCh}}$, we have that $w \notin \mathsf{BadCh}(\tau|_w)$. Therefore, $\mathsf{SZ}(f(W), w)$ is `false` where f is as defined in $\mathsf{CheckBad}(\tau', w)$. Since we have here that $f(w) = 0$, the polynomial $f(W)$ must be the zero polynomial. In particular its W term must be zero, i.e.,

$$\hat{t} = \Big\langle \big(a_{I\mathbf{g}} x + a_{O\mathbf{g}} x^2 + \mathbf{y}^{-n} \circ (\mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_R) x + s_{\mathbf{g}} x^3\big), \mathbf{y}^n \circ \big(a_{I\mathbf{h}} x + a_{O\mathbf{h}} x^2 - \mathbf{1}^n + \mathbf{y}^{-n} \circ (\mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_L) x + \mathbf{y}^{-n} \circ (\mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_O) + s_{\mathbf{g}} x^3\big) \Big\rangle \ .$$

Plugging in the value of $\hat{t}$ obtained in (13), we have that

$$x^2(\delta(y,z) + \langle \mathbf{z}^{Q+1}_{[1:]}, \mathbf{c} \rangle) + t_{1g} x + \sum_{i=3}^{6} t_{ig} x^i - \Big\langle a_{I\mathbf{g}} x + a_{O\mathbf{g}} x^2 + \mathbf{y}^{-n} \circ (\mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_R) x + s_{\mathbf{g}} x^3, \mathbf{y}^n \circ \big(a_{I\mathbf{h}} x + a_{O\mathbf{h}} x^2 - \mathbf{1}^n + \mathbf{y}^{-n} \circ (\mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_L) x + \mathbf{y}^{-n} \circ (\mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_O) + s_{\mathbf{g}} x^3\big) \Big\rangle = 0 \ .$$

Since $\tau \notin \mathcal{T}^{\mathsf{ACSPf}}_{\mathsf{BadCh}}$, we have that $x \notin \mathsf{BadCh}(\tau|_x)$. Therefore, $\mathsf{SZ}(f_4(X), x)$ is `false` where f_4 is as defined in $\mathsf{CheckBad}(\tau', x)$. Since we have here that $f_4(x) = 0$, the polynomial $f_4(X)$ must be the zero polynomial. In particular its X^2 term must be zero, i.e.,

$$\begin{aligned} &\delta(y,z) + \langle \mathbf{z}^{Q+1}_{[1:]}, \mathbf{c} \rangle - \langle a_{I\mathbf{g}}, \mathbf{y}^n \circ a_{I\mathbf{h}} \rangle - \langle a_{I\mathbf{g}}, \mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_L \rangle + \langle a_{O\mathbf{g}}, \mathbf{y}^n \rangle \\ &- \langle \mathbf{y}^{-n} \circ (\mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_R), \mathbf{y}^n \circ a_{I\mathbf{h}} \rangle - \langle \mathbf{y}^{-n} \circ (\mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_R), (\mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_L) \rangle = 0 \ . \end{aligned}$$

Plugging in $\delta(y,z) = \langle \mathbf{y}^{-n} \circ (\mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_R), (\mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_L) \rangle$, we get

$$\begin{aligned} &\langle \mathbf{z}^{Q+1}_{[1:]}, \mathbf{c} \rangle - \langle a_{I\mathbf{g}}, \mathbf{y}^n \circ a_{I\mathbf{h}} \rangle - \langle a_{I\mathbf{g}}, \mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_L \rangle + \langle a_{O\mathbf{g}}, \mathbf{y}^n \rangle - \langle \mathbf{y}^{-n} \circ (\mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_R), \mathbf{y}^n \circ a_{I\mathbf{h}} \rangle \\ &- \langle a_{O\mathbf{g}}, \mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_O \rangle = 0 \ . \end{aligned}$$

Simplifying and rearranging we get

$$\langle \mathbf{z}^{Q+1}_{[1:]}, \mathbf{c} - \mathbf{W}_L \cdot a_{I\mathbf{g}} - \mathbf{W}_R \cdot a_{I\mathbf{h}} - \mathbf{W}_O \cdot a_{O\mathbf{g}} \rangle - \langle a_{I\mathbf{g}} \circ a_{I\mathbf{h}} - a_{O\mathbf{g}}, \mathbf{y}^n \rangle = 0 \ .$$

Since $\tau \notin \mathcal{T}^{\mathsf{ACSPf}}_{\mathsf{BadCh}}$, we have that $(y,z) \notin \mathsf{BadCh}(\tau|_{(y,z)})$. Therefore, $\mathsf{SZ}(f(Y,Z),(y,z))$ is `false` where f is as defined in $\mathsf{CheckBad}(\tau',(y,z))$. Since we have here that $f(y,z)=0$, the polynomial $f(Y,Z)$ is the vector polynomial. Equating all its coefficients to zero, we get

$$\mathbf{W}_L \cdot a_{I\mathbf{g}} + \mathbf{W}_R \cdot a_{I\mathbf{h}} + \mathbf{W}_O \cdot a_{O\mathbf{g}} = \mathbf{c}\ , a_{I\mathbf{g}} \circ a_{I\mathbf{h}} = a_{O\mathbf{g}}\ .$$

Since $(\mathbf{a}^*_L, \mathbf{a}^*_R, \mathbf{a}^*_O)$ returned by e is $(a_{I\mathbf{g}}, a_{I\mathbf{h}}, a_{O\mathbf{g}})$ we have that

$$\mathbf{a}^*_L \circ \mathbf{a}^*_R = \mathbf{a}^*_O \text{ and } \mathbf{W}_L \cdot \mathbf{a}^*_L + \mathbf{W}_R \cdot \mathbf{a}^*_R + \mathbf{W}_O \cdot \mathbf{a}^*_O = \mathbf{c}\ .$$

□

5.4 Proof of Lemma 8

From the statement of the algebraic lemma, it is evident that we need to eliminate everything except for $p'_{\mathbf{g}}, p'_{\mathbf{h}}, y, p'_u, w$ to obtain a relation between them. Our first step would be to plug in the values of a, b from the first two sets of equalities into the third – this would eliminate a, b. Then we shall exploit the first two sets of equalities and the definition of Bad to arrive at an equation solely in terms of $p'_{\mathbf{g}}, p'_{\mathbf{h}}, y, p'_u, w$.

Proof (Lemma 8).

First we observe that given that $\mathsf{Bad}(\mathsf{params}, x, j) \neq \mathtt{true}$, if for any of the polynomials $p(X)$ on which SZ is called in Bad, $p(x)$ is zero, then the polynomial $p(X)$ is the zero polynomial. We shall use this observation repeatedly in this proof.

SIMPLIFYING NOTATION. We introduce some new notation for simplicity. We define the following polynomials. For all $k \in \{1,\ldots,\log n\}$, for all $j \in \{0,\ldots,n-1\}$

$$\begin{aligned} f^{\mathbf{g}}_{k,j}(X) &= l_{kg_{1+j}} X^2 + r_{kg_{1+j}} X^{-2} + p'_{g_{1+j}} + \sum_{i=1}^{k-1} (l_{ig_{1+j}} x_i^2 + r_{ig_{1+j}} x_i^{-2})\ , \\ f^{\mathbf{h}}_{k,j}(X) &= l_{kh_{1+j}} X^2 + r_{kh_{1+j}} X^{-2} + p'_{h_{1+j}} + \sum_{i=1}^{k-1} (l_{ih_{1+j}} x_i^2 + r_{ih_{1+j}} x_i^{-2})\ . \end{aligned} \tag{18}$$

For all $k \in \{1,\ldots,\log n\}$

$$f^u_k(X) = l_{ku} X^2 + r_{ku} X^{-2} + p'_u + \sum_{i=1}^{k-1} (l_{iu} x_i^2 + r_{iu} x_i^{-2})\ . \tag{19}$$

Using our notation in (18) and (19), we can re-write our given equalities as

1. for $k = 0,\ldots,n-1$
$$a = f^{\mathbf{g}}_{\log n,k}(x_{\log n}) \cdot \left(\prod_{i=1}^{\log n} x_i^{(-1)^{\mathsf{bit}(k,i,\log n)}} \right)\ .$$
2. for $k = 0,\ldots,n-1$
$$b = f^{\mathbf{h}}_{\log n,k}(x_{\log n}) \cdot y^{((k))} \cdot \left(\prod_{i=1}^{\log n} x_i^{(-1)^{1-\mathsf{bit}(k,i,\log n)}} \right)\ .$$
3. $$f^u_{\log n}(x_{\log n}) - w \cdot ab = 0\ .$$

ELIMINATING a, b IN THE THIRD EQUALITY. First off, we plug the values of a, b we obtain for $k = 0$ into the third equality. We obtain

$$f^{u}_{\log n}(x_{\log n}) - w \cdot f^{\mathbf{g}}_{\log n,1}(x_{\log n}) \cdot f^{\mathbf{h}}_{\log n,1}(x_{\log n}) = 0 \ . \tag{20}$$

In order to eliminate all variable except $p'_{\mathbf{g}}, p'_{\mathbf{h}}, y, p'_u, w$, we need to use the first two sets of equalities to obtain relations that we can plug back into (20).

RELATIONS FROM THE FIRST SET OF EQUALITIES. The first set of equalities gave us that for $k = 0, \ldots, n-1$

$$a = f^{\mathbf{g}}_{\log n,k}(x_{\log n}) \cdot \left(\prod_{i=1}^{\log n} x_i^{(-1)^{\mathsf{bit}(k,i,\log n)}} \right) \ . \tag{21}$$

Let $t \in \{1, \ldots, \log n\}$ and $j \in \{0, \ldots, n/2^t - 1\}$. Equating the values of a for $k = j$ and $k = j + n/2^t$, we get

$$f^{\mathbf{g}}_{\log n,j}(x_{\log n}) \cdot \left(\prod_{i=1}^{\log n} x_i^{(-1)^{\mathsf{bit}(j,i,\log n)}} \right) = f^{\mathbf{g}}_{\log n,j+n/2^t}(x_{\log n}) \cdot \left(\prod_{i=1}^{\log n} x_i^{(-1)^{\mathsf{bit}(j+n/2^t,i,\log n)}} \right) \ .$$

Since $j \in \{0, \ldots, n/2^t - 1\}$, j and $j + n/2^t$ differ only in the t^{th} bit. So, we have for $t \in \{1, \ldots, \log n\}$, $j \in \{0, \ldots, n/2^t - 1\}$

$$f^{\mathbf{g}}_{\log n,j}(x_{\log n}) \cdot x_t^2 = f^{\mathbf{g}}_{\log n,j+n/2^t}(x_{\log n}) \ . \tag{22}$$

We shall next show that for all $t \in \{1, \ldots, \log n\},$ for all $j \in \{0, \ldots, n/2^t - 1\}$

$$l_{tg_{1+j}} = 0 \ , r_{tg_{1+j}} = f^{\mathbf{g}}_{t-1,j+n/2^t}(x_{t-1}) \ .$$

First we show it for $t = \log n$- in this case j can take the value only 0. We have that

$$f^{\mathbf{g}}_{\log n,0}(x_{\log n}) \cdot x^2_{\log n} - f^{\mathbf{g}}_{\log n,1}(x_{\log n}) = 0 \ .$$

Since $\mathsf{Bad}(\mathsf{params}, x_{\log n}, \log n) = \texttt{false}$

$$f^{\mathbf{g}}_{\log n,0}(X) \cdot X^2 - f^{\mathbf{g}}_{\log n,1}(X)$$

is the zero polynomial. Equating the constant term to 0 we get

$$r_{(\log n)g_1} = f^{\mathbf{g}}_{\log n-1,1}(x_{\log n-1}) \ ,$$

Equating the X^4 term to 0 we get,

$$l_{(\log n)g_1} = 0 \ .$$

Hence, it holds for $t = \log n$. Now let $t = t' < \log n$. We have that for $j \in \{0, \ldots, n/2^{t'} - 1\}$.

$$f^{\mathbf{g}}_{\log n,j}(x_{\log n}) \cdot x^2_{t'} - f^{\mathbf{g}}_{\log n,j+n/2^{t'}}(x_{\log n}) = 0 \ .$$

Since $\mathsf{Bad}(\mathsf{params}, x_{\log n}, \log n) = \texttt{false}$

$$f^{\mathbf{g}}_{\log n,j}(X) \cdot x^2_{t'} - f^{\mathbf{g}}_{\log n,j+n/2^{t'}}(X)$$

is the zero polynomial. Therefore, its constant term is 0, i.e.,

$$f^{\mathbf{g}}_{\log n-1,j}(x_{\log n-1}) \cdot x_{t'}^2 - f^{\mathbf{g}}_{\log n,j+n/2^{t'}}(x_{\log n-1}) = 0 \ .$$

Using similar series of arguments (since for all $j \in \{\log n-1, \log n-2, \ldots, t'\}$: $\mathsf{Bad}(\mathsf{params}, x_j, j) = \texttt{false}$) we can arrive at

$$f^{\mathbf{g}}_{t',j}(x_{t'}) \cdot x_{t'}^2 - f^{\mathbf{g}}_{t',j+n/2^{t'}}(x_{t'}) = 0 \ .$$

Now, since $\mathsf{Bad}(\mathsf{params}, x'_t, t') = \texttt{false}$

$$f^{\mathbf{g}}_{t',j}(X) \cdot X^2 - f^{\mathbf{g}}_{t',j+n/2^{t'}}(X)$$

must be the zero polynomial. Equating the constant term to 0 we get for $t' > 1$

$$r_{t'g_j} = f^{\mathbf{g}}_{t'-1,j+n/2^{t'}}(x_{t'-1}) \ ,$$

and for $t' = 1$

$$r_{1g_{1+j}} = p'_{g_{1+j+n/2}} \ .$$

Equating the X^4 term to 0 we get,

$$l_{t'g_j} = 0 \ .$$

Hence for all $t \in \{2, \ldots, \log n\},$ for all $j \in \{0, \ldots, n/2^t - 1\}$

$$l_{tg_{1+j}} = 0 \ , r_{tg_{1+j}} = f^{\mathbf{g}}_{t-1,j+n/2^t}(x_{t-1}) \ , \tag{23}$$

and for all $j \in \{0, \ldots, n/2 - 1\}$

$$r_{1g_{1+j}} = p'_{g_{1+j+n/2}} \ , l_{1g_{1+j}} = 0 \ . \tag{24}$$

RELATIONS FROM THE SECOND SET OF EQUALITIES. Now, we can go through an identical process for the second set of equalities and obtain that (we omit the calculations since they are identical to the ones we saw previously)

1. for all $t \in \{2, \ldots, \log n\},$ for all $j \in \{0, \ldots, n/2^t - 1\}$

$$r_{th_{1+j}} = 0 \ , l_{th_{1+j}} = f^{\mathbf{h}}_{t-1,j+n/2^t}(x_t) \cdot y^{n/2^t} \ . \tag{25}$$

2. for all $j \in \{0, \ldots, n/2 - 1\}$

$$r_{1h_{1+j}} = 0 \ , l_{1h_{1+j}} = p'_{h_{1+j+n/2}} \cdot y^{n/2} \ . \tag{26}$$

PUTTING IT ALL TOGETHER. Finally, we are ready to use the obtained relations. We shall show using induction on k that for all $k \in \{1, \ldots, \log n\}$

$$f^u_k(x_k) - w \cdot \sum_{j=0}^{n/2^k-1} f^{\mathbf{g}}_{k,j}(x_k) \cdot f^{\mathbf{h}}_{k,j}(x_k) \cdot y^j = 0 \ .$$

The base case for $k = \log n$ is true since (20) holds.

Now assuming it holds for some $k = k'$ we shall show that it holds for $k' - 1$ as well. Using induction hypothesis we have that

$$f^u_{k'}(x_{k'}) - w \cdot \sum_{j=0}^{n/2^{k'}-1} f^{\mathbf{g}}_{k',j}(x_{k'}) \cdot f^{\mathbf{h}}_{k',j}(x_{k'}) \cdot y^j = 0 \ .$$

Since $\mathsf{Bad}(\mathsf{params}, x'_k, k') \neq \mathtt{true}$, the polynomial

$$f^u_{k'}(X) - w \cdot \sum_{j=0}^{n/2^{k'}-1} f^{\mathbf{g}}_{k',j}(X) \cdot f^{\mathbf{h}}_{k',j}(X) \cdot y^j$$

must be the zero polynomial, i.e., in particular its constant term is zero. It's constant term can be written as

$$f^u_{k'-1}(x_{k'-1}) - w \cdot \sum_{j=0}^{n/2^{k'}-1} f^{\mathbf{g}}_{k'-1,j}(x_{k'-1}) \cdot f^{\mathbf{h}}_{k'-1,j}(x_{k'-1}) \cdot y^j$$
$$- w \cdot \sum_{j=0}^{n/2^{k'}-1} (l_{k'g_{1+j}} \cdot r_{k'h_{1+j}} + r_{k'g_{1+j}} \cdot l_{k'h_{1+j}}) \cdot y^j) \ .$$

From (23) and (25) we have that for $j \in \{0, \ldots, n/2^{k'} - 1\}$

$$r_{k'g_{1+j}} = f^{\mathbf{g}}_{k'-1,j+n/2^{k'}}(x_{k'-1}) \ , \ l_{k'g_{1+j}} = 0 \ , \ r_{k'h_{1+j}} = 0 \ , \ l_{k'h_{1+j}} = f^{\mathbf{h}}_{k'-1,j+n/2^{k'}}(x_{k'-1}) \cdot y^{n/2^{k'}} \ .$$

So, equating the constant term to 0 we have that

$$f^u_{k'-1}(x_{k'-1}) - w \cdot \sum_{j=0}^{n/2^{k'}-1} (f^{\mathbf{g}}_{k'-1,j}(x_{k'-1}) \cdot f^{\mathbf{h}}_{k'-1,j}(x_{k'-1}) \cdot y^j)$$
$$- w \cdot \sum_{j=0}^{n/2^{k'}-1} ((f^{\mathbf{g}}_{k'-1,j+n/2^{k'}}(x_{k'-1}) \cdot f^{\mathbf{h}}_{k'-1,j+n/2^{k'}}(x_{k'-1})) \cdot y^{j+n/2^{k'}}) = 0 \ .$$

This can be simplified to

$$f^u_{k'-1}(x_{k'-1}) - w \cdot \sum_{j=0}^{n/2^{k'-1}-1} (f^{\mathbf{g}}_{k'-1,j}(x_{k'-1}) \cdot f^{\mathbf{h}}_{k'-1,j}(x_{k'-1}) \cdot y^j = 0 \ .$$

Hence we have shown that it holds for $k = k' - 1$. Hence, by induction we arrive at

$$f^u_1(x_1) - w \cdot \sum_{j=0}^{n/2-1} (f^{\mathbf{g}}_{1,j}(x_1) \cdot f^{\mathbf{h}}_{1,j}(x_1) \cdot y^j = 0 \ .$$

Since $\mathsf{Bad}(\mathsf{params}, x'_k, k') \neq \mathtt{true}$, the polynomial

$$f^u_1(X) - w \cdot \sum_{j=0}^{n/2-1} (f^{\mathbf{g}}_{1,j}(X) \cdot f^{\mathbf{h}}_{1,j}(X) \cdot y^j)$$

is the zero polynomial, i.e., in particular its constant term is 0. So, we have that

$$p'_u - w \sum_{j=0}^{n/2-1} p'_{g_{1+j}} \cdot p'_{h_{1+j}} \cdot y^j - w \sum_{j=0}^{n/2-1} (l_{1g_{1+j}} \cdot r_{1h_{1+j}} + r_{1g_{1+j}} \cdot l_{1h_{1+j}}) \cdot y^j = 0 \,.$$

From (24) and (26) we have that for $j \in \{0, \ldots, n/2-1\}$

$$r_{1g_{1+j}} = p'_{g_{1+j+n/2}} \,, l_{1g_{1+j}} = 0, r_{1h_{1+j}} = 0 \,, l_{1h_{1+j}} = p'_{h_{1+j+n/2}} \cdot y^{n/2} \,.$$

So, we have that

$$p'_u - w \sum_{j=0}^{n/2-1} p'_{g_{1+j}} \cdot p'_{h_{1+j}} \cdot y^j - w \sum_{j=0}^{n/2-1} (l_{1g_{1+j}} \cdot r_{1h_{1+j}} + r_{1g_{1+j}} \cdot l_{1h_{1+j}} \cdot y^{n/2}) \cdot y^j = 0 \,.$$

Simplifying we get that

$$p'_u = w \cdot \langle p'_{\mathbf{g}}, p'_{\mathbf{h}} \circ \mathbf{y}^n \rangle \,.$$

□

6 Online srs-wee Security of Sonic

We apply our framework to prove srs-wee security of Sonic [MBKM19] which is an interactive argument for arithmetic circuit satisfiability based on pairings (we refer to this argument as SnACSPf). The argument SnACSPf is again an argument of knowledge for the relation (11). Sonic represents arithmetic circuits using the same constraint system as the one used in Bulletproofs. The constraint system has 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 $\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O$, with additional $Q \leqslant 2n$ linear constraints. The linear constraints can be represented as $\mathbf{a}_L \cdot \mathbf{W}_L + \mathbf{a}_R \cdot \mathbf{W}_R + \mathbf{a}_O \cdot \mathbf{W}_O = \mathbf{c}$, where $\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O \in \mathbb{Z}_p^{Q \times n}$.

PAIRINGS. As stated before, SnACSPf is based on pairings. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be groups of prime order p with generators $g \in \mathbb{G}_1, h \in \mathbb{G}_2$. A pairing is a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ such that $e(g^a, h^b) = e(g,h)^{ab}$ for all $a, b \in \mathbb{Z}_p$ and $e(g,h)$ is a generator of $\mathbb{G}_T$. In our AGM analysis, we shall consider symmetric pairings, i.e., $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$. We shall assume that SnACSPf $=$ SnACSPf$[\mathbb{G}, \mathbb{G}_T, e]$ is instantiated on the understood families of groups $\mathbb{G} = \{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}^+}$ (with order $p = p(\lambda)$) and $\mathbb{G}_T = \{\mathbb{G}_{T,\lambda}\}_{\lambda \in \mathbb{N}^+}$ such that there exists a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

The interactive argument SnACSPf. is a argument of knowledge for the following relation.

$$\begin{aligned} R = \{ \big(((n, Q \in \mathbb{N}^+), (\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O \in \mathbb{Z}_p^{Q \times n}, \mathbf{c} \in \mathbb{Z}_p^Q)), (\mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O \in \mathbb{Z}_p^n) \big) : \\ \mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O \wedge \mathbf{W}_L \cdot \mathbf{a}_L + \mathbf{W}_R \cdot \mathbf{a}_R + \mathbf{W}_O \cdot \mathbf{a}_O = \mathbf{c} \} \,. \end{aligned}$$

The setup algorithm SnACSPf.Setup fixes an integer d such that $4n > d > 3n$. It generates the bilinear parameters $\mathsf{bp} = ((p, \mathbb{G}, \mathbb{G}_T, e, g, h))$. It then samples α, x uniformly at random from $\mathbb{Z}_p$. It sets

$$\mathsf{srs} = \{g, \{g^{x^i}\}_{i=-d}^{d}, \{h^{x^i}\}_{i=-d}^{d}, \{h^{\alpha x^i}\}_{i=-d}^{d}, \{g^{\alpha x^i}\}_{\substack{i=-d \\ i \neq 0}}^{d}, e(g, h^\alpha)\} \,.$$

It returns $(\mathsf{bp}, \mathsf{srs})$ as its output.

As shown in [MBKM19] the argument for the above relation proceeds by defining following polynomials $r(X,Y), k(Y), s(X,Y), t(X,Y)$ and proving that the constant term of $t(X,Y)$ is zero.

$$
\begin{aligned}
r(X,Y) &\leftarrow \langle \mathbf{a}_L, \mathbf{X}_{[1:]}^{n+1} \circ \mathbf{Y}_{[1:]}^{n+1} \rangle + \langle \mathbf{a}_R, \mathbf{X}_{[1:]}^{-n-1} \circ \mathbf{Y}_{[1:]}^{n+1} \rangle + \langle \mathbf{a}_O, \mathbf{X}_{[n+1:]}^{-2n-1} \circ \mathbf{Y}_{[n+1:]}^{-2n-1} \rangle \,, \\
k(Y) &\leftarrow \langle \mathbf{c}, \mathbf{Y}_{[n+1:]}^{Q+n+1} \rangle \,, \\
s(X,Y) &\leftarrow \mathbf{Y}_{[n+1:]}^{Q+n+1} \cdot (\mathbf{W}_L \cdot \mathbf{X}_{[1:]}^{-n-1} + \mathbf{W}_R \cdot \mathbf{X}_{[1:]}^{n+1} + \mathbf{W}_O \cdot \mathbf{X}_{[n+1:]}^{2n+1}) + \langle -\mathbf{Y}_{[1:]}^{n+1} - \mathbf{Y}_{[1:]}^{-n-1}, \mathbf{X}_{[n+1:]}^{2n+1} \rangle \,, \\
t(X,Y) &\leftarrow r(X,1)(r(X,Y) + s(X,Y)) - k(Y) \,.
\end{aligned}
$$

Note that the verifier can evaluate $s(X,Y), k(Y)$ without the witness. However evaluating $s(X,Y)$ is expensive, hence the prover the prover computes the value and the prover and the verifier engage in an argument for signature of correct computation where the prover demonstrates to the verifier that it sent the correct evaluation. This argument for signature of correct computation assumes that the polynomial $s(X,Y)$ can be expressed as a sum of M polynomials of the form $\psi_j(X,Y) = \sum_{i=1}^{n} \psi_{j,\sigma_{j,i}} X^i Y^{\sigma_{j,i}}$ where $\sigma_j = (\sigma_{j,1}, \ldots, \sigma_{j,n})$ is a permutation of $(1, \ldots, n)$. As stated in [MBKM19], for any given arithmetic circuit, one can devise a constraint system such that $s(X,Y)$ can be represented as a sum of $M = O(1)$ such polynomials.

The prover and the verifier algorithms, SnACSPf.P, SnACSPf.V are shown in Figure 16 with all sub-components defined in Figures 17 to 21. The complexity of the protocol necessitates this modular description. Figure 17 describes the polynomial commitments used in Sonic and Figure 18 describes the signature of correct computation which uses the polynomial permutation argument defined in Figure 19 which in turn uses the grand product argument in Figure 20. The argument for well-formedness of commitments used by the grand product argument is defined in Figure 21.

In the soundness analysis of SnACSPf in [MBKM19], only the bounded polynomial extractibility and evaluation binding of the commitment scheme is analysed in the AGM.[6] Here we give an analysis of the srs-wee of SnACSPf in the AGM.

We prove the following theorem that establishes an upper bound on the online srs-wee security of SnACSPf.

Theorem 7. *Let $\mathbb{G} = \{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}^+}$ be a family of groups with order $p = p(\lambda)$. Let $\mathbb{G}_T = \{\mathbb{G}_{T,\lambda}\}_{\lambda \in \mathbb{N}^+}$ be a family of groups such that there exists a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Let $\mathsf{SnACSPf} = \mathsf{SnACSPf}[\mathbb{G}, \mathbb{G}_T, e]$ be the interactive argument as described in Figure 16, for the relation R in (11). We can construct an extractor $\mathcal{E}$ such that for any non-uniform algebraic prover $\mathcal{P}_{\mathsf{alg}}$ making at most $q = q(\lambda)$ queries to its oracle, there exist non-uniform adversaries $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$ with the property that for any (computationally unbounded) distinguisher $\mathcal{D}$, for all $\lambda \in \mathbb{N}^+$*

$$
\mathsf{Adv}^{\mathsf{sr\text{-}wee}}_{\mathsf{SnACSPf},R}(\mathcal{P}_{\mathsf{alg}}, \mathcal{D}, \mathcal{E}, \lambda) \leqslant \frac{18nq}{p-1} + \mathsf{Adv}^{4n\text{-}\mathsf{dl}}_{\mathbb{G}}(\mathcal{F}_1, \lambda) + \mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{F}_2, \lambda) + \mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{F}_3, \lambda) \,.
$$

Moreover, the time complexities of the extractor $\mathcal{E}$ and adversaries $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$ are all $O(q \cdot n)$.

We can show that the bound in Theorem 7 is tight by constructing a cheating prover like we did in Theorem 5. Using Theorem 2, we get a corollary about fs-ext-1 security of $\mathsf{FS}^{\mathbf{RO}}[\mathsf{SnACSPf}]$.

Corollary 3. *Let $\mathbb{G} = \{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}^+}$ be a family of groups with order $p = p(\lambda)$. Let $\mathbb{G}_T = \{\mathbb{G}_{T,\lambda}\}_{\lambda \in \mathbb{N}^+}$ be a family of groups such that there exists a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Let $\mathsf{SnACSPf} = \mathsf{SnACSPf}[\mathbb{G}, \mathbb{G}_T, e]$*

[6] The reduction of bounded polynomial extractibility to the variant of q-dl defined in the paper does not seem to account for the fact that an algebraic adversary can represent the commitments in terms of powers of h as well.

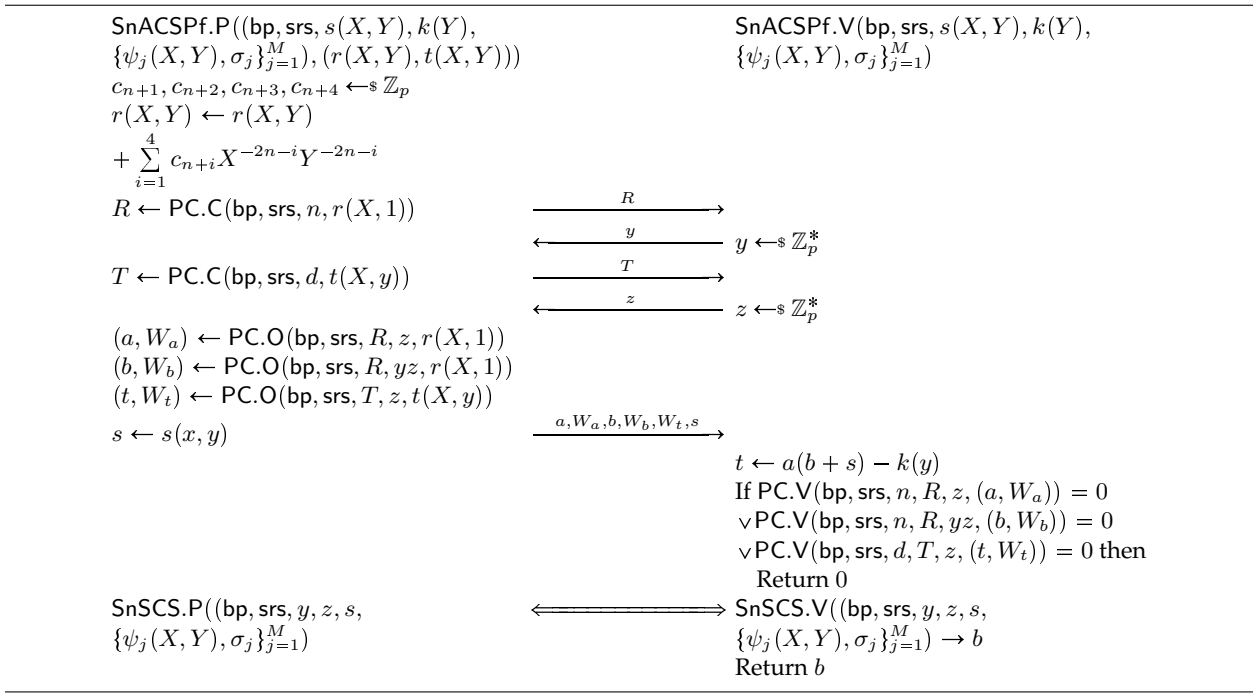

Fig. 16. The interactive argument for arithmetic circuit satisfiability in Sonic.

PC.C(bp, srs, $n, f(X)$)
$F \leftarrow g^{\alpha x^{d-n}}$
Return F

PC.O(bp, srs, $F, z, f(X)$)
$w(X) \leftarrow \frac{f(X)-f(z)}{X-z}$
$W \leftarrow g^{w(x)}$
Return $(f(z), W)$

PC.V(bp, srs, $n, F, z, (v, W)$)
$F \leftarrow g^{\alpha x^{d-n}}$
If $e(W, h^{\alpha x}) e(g^v W^{-z}, h^\alpha) = e(F, h^{x^{-d+n}})$ then
Return 0
Return 1

Fig. 17. Polynomial commitments in Sonic

$\mathsf{SnSCS.P}(\mathsf{bp}, \mathsf{srs}, y, z, s,$
$, \{\psi_j(X,Y), \sigma_j\}_{j=1}^{M})$
For $j = 1$ to M do
  $P_1 \leftarrow \mathsf{PC.C}(\mathsf{bp}, \mathsf{srs}, d, \sum_{i=1}^{n} X^i)$
  $P_2 \leftarrow \mathsf{PC.C}(\mathsf{bp}, \mathsf{srs}, d, \sum_{i=1}^{n} \psi_{j,i} X^i)$
  $P_3 \leftarrow \mathsf{PC.C}(\mathsf{bp}, \mathsf{srs}, d, \sum_{i=1}^{n} iX^i)$
  $P_4 \leftarrow \mathsf{PC.C}(\mathsf{bp}, \mathsf{srs}, d, \sum_{i=1}^{n} \sigma_{j,i} X^i)$
  $\mathsf{SnPP.P}((\mathsf{bp}, \mathsf{srs}, P_1, P_2, P_3, P_4, y, z), (\psi_i(X,Y), \sigma_i))$ $\Longleftrightarrow$ $\mathsf{SnPP.V}(\mathsf{bp}, \mathsf{srs}, P_1, P_2, P_3, P_4, y, z) \rightarrow (b_j, s_j)$
  If $b_j = 0$ then return 0
If $s = \sum_{j=1}^{M} s_j$ then return 1
Return 0

$\mathsf{SnSCS.V}(\mathsf{bp}, \mathsf{srs}, y, z, s,$
$\{\psi_j(X,Y), \sigma_j\}_{j=1}^{M})$
For $i = 1$ to M do
  $P_1 \leftarrow \mathsf{PC.C}(\mathsf{bp}, \mathsf{srs}, d, \sum_{i=1}^{n} X^i)$
  $P_2 \leftarrow \mathsf{PC.C}(\mathsf{bp}, \mathsf{srs}, d, \sum_{i=1}^{n} \psi_{j,i} X^i)$
  $P_3 \leftarrow \mathsf{PC.C}(\mathsf{bp}, \mathsf{srs}, d, \sum_{i=1}^{n} iX^i)$
  $P_4 \leftarrow \mathsf{PC.C}(\mathsf{bp}, \mathsf{srs}, d, \sum_{i=1}^{n} \sigma_{j,i} X^i)$

Fig. 18. The signature of correct computation in Sonic.

$\mathsf{SnPP.P}((\mathsf{bp}, \mathsf{srs}, P_1, P_2, P_3, P_4, y, z),$
$, \psi(X,Y), \phi(X,Y), \sigma_i)$
$S \leftarrow \mathsf{PC.C}(\mathsf{bp}, \mathsf{srs}, d, \psi(X,y))$
$S' \leftarrow \mathsf{PC.C}(\mathsf{bp}, \mathsf{srs}, d, \phi(X,y))$ $\xrightarrow{S,S'}$
$\xleftarrow{\delta,\beta,\gamma}$ $\delta, \beta, \gamma \leftarrow_{\$} \mathbb{Z}_p^*$
$U \leftarrow SP_4^{\beta} P_1^{\gamma}; V \leftarrow S' P_3^{\beta} P_1^{\gamma}$
$(s, W) \leftarrow \mathsf{PC.O}(\mathsf{bp}, \mathsf{srs}, S, z, \psi(X,y))$
$(v, W') \leftarrow \mathsf{PC.O}(\mathsf{bp}, \mathsf{srs}, S', \delta, \phi(X,y))$
$(v, Q') \leftarrow \mathsf{PC.O}(\mathsf{bp}, \mathsf{srs}, P_2, \delta y, \sum_{i=1}^{n} \psi_i X^i)$
$u(X) \leftarrow \sum_{i=1}^{n} \psi_{\sigma_i} y^{\sigma_i} X^i + \beta \sigma_i X^i + \gamma X^i$
$v(X) \leftarrow \sum_{i=1}^{n} \psi_i y^i X^i + \beta i X^i + \gamma X^i$ $\xrightarrow{s,W,v,W',Q'}$
$U \leftarrow SP_4^{\beta} P_1^{\gamma}; V \leftarrow S' P_3^{\beta} P_1^{\gamma}$
If $\mathsf{PC.V}(\mathsf{bp}, \mathsf{srs}, d, S, z, (s, W)) = 0$
$\vee \mathsf{PC.V}(\mathsf{bp}, \mathsf{srs}, d, S', \delta, (v, W')) = 0$
$\vee \mathsf{PC.V}(\mathsf{bp}, \mathsf{srs}, d, P_2, \delta y, (v, Q')) = 0$
  Return $(0, \perp)$
$\mathsf{SnGP.P}((\mathsf{bp}, \mathsf{srs}, U, V), (u(X), v(X)))$ $\Longleftrightarrow$ $\mathsf{SnGP.V}(\mathsf{bp}, \mathsf{srs}, U, V) \rightarrow b$
If $b = 0$ then return $(0, \perp)$
Return $(1, s)$

Fig. 19. The polynomial permutation argument in Sonic.

be the interactive argument as described in Figure 16, for the relation R in (11). Let $\mathsf{FS}^{\mathbf{RO}}[\mathsf{SnACSPf}]$ be the non-interactive argument obtained by applying the Fiat-Shamir transform to $\mathsf{SnACSPf}$ using a random oracle. We can construct an extractor $\mathcal{E}$ such that for any non-uniform algebraic prover $\mathcal{P}_{\mathsf{alg}}$ making at most $q = q(\lambda)$ queries to the random oracle there exist non-uniform adversaries $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$ with the property that for all $\lambda \in \mathbb{N}^+$

$$\mathsf{Adv}^{\mathsf{fs\text{-}ext\text{-}1}}_{\mathsf{FS}^{\mathbf{RO}}[\mathsf{SnACSPf}],R}(\mathcal{P}_{\mathsf{alg}}, \mathcal{E}, \lambda) \leqslant \frac{18nq + q + 1}{p - 1} + \mathsf{Adv}^{4n\text{-}\mathsf{dl}}_{\mathbb{G}}(\mathcal{F}_1, \lambda) + \mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{F}_2, \lambda) + \mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{F}_3, \lambda) \ .$$

Moreover, the time complexities of the extractor $\mathcal{E}$ and adversaries $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$ are all $O(q \cdot n)$.

Additionally, using techniques similar to those in the proof of Theorem 7, we can prove a similar bound for fs-ext-2 security of $\mathsf{FS}^{\mathbf{RO}}[\mathsf{SnACSPf}]$.

$\mathsf{SnGP.P}((\mathsf{bp}, \mathsf{srs}, U, V),$ $\mathsf{SnGP.V}(\mathsf{bp}, \mathsf{srs}, U, V)$

$(u(X) = \sum_{i=1}^{n} u_i X^i, v(X) = \sum_{i=1}^{n} v_i X^i))$

$(a_1, \ldots, a_n) \leftarrow (u_1, \ldots, u_n)$

$(a_{n+2}, \ldots, a_{2n+1}) \leftarrow (v_1, \ldots, v_n)$

$a_{n+1} \leftarrow (\prod_{i=1}^{n} a_i)^{-1}$

$c_1 \leftarrow a_1$

For $i = 2$ to $2n + 1$ do

  $c_i \leftarrow c_{i-1} \cdot a_i$

$c(X) \leftarrow \sum_{i=1}^{2n+1} c_i X^i; a(X) \leftarrow \sum_{i=1}^{2n+1} a_i X^i$

$r(X, Y) \leftarrow Y(\sum_{i=1}^{2n+1} a_i X^i Y^i$

$+a_{n+1} X^{n+1} Y^{n+1})$

$s(X, Y) \leftarrow X^{n+2} + X^{n+1} Y - X^{2n+2} Y$

$r'(X, Y) \leftarrow X^{-1} + \sum_{i=1}^{n} c_i X^{-i-1}$

$k(Y) \leftarrow 1 + \sum_{i=1}^{2n+1} c_i Y^{i+1}$

$t(X, Y) \leftarrow (r(X, Y) + s(X, Y)) r'(X, Y)$

$-k(Y)$

$A \leftarrow g^{a_{n+1} \alpha x^{n+1}} U V^{x^{n+1}}$

$C \leftarrow \mathsf{PC.C}(\mathsf{bp}, \mathsf{srs}, d, c(X))$

$C_w \leftarrow \mathsf{SnWF.Prv}(\mathsf{bp}, \mathsf{srs}, 2n + 1, c(X))$

$U_w \leftarrow \mathsf{SnWF.Prv}(\mathsf{bp}, \mathsf{srs}, n, u(X))$

$V_w \leftarrow \mathsf{SnWF.Prv}(\mathsf{bp}, \mathsf{srs}, n, v(X))$ $\xrightarrow{A, C, C_w, U_w, V_w, a_{n+1}}$

$\xleftarrow{y}$ $y \leftarrow\$ \mathbb{Z}_p^*$

$T \leftarrow \mathsf{PC.C}(\mathsf{bp}, \mathsf{srs}, d, t(X, y))$ $\xrightarrow{T}$

$\xleftarrow{z}$ $z \leftarrow\$ \mathbb{Z}_p^*$

$(a, W_a) \leftarrow \mathsf{PC.O}(\mathsf{bp}, \mathsf{srs}, A, yz, a(X))$

$(c, W_c) \leftarrow \mathsf{PC.O}(\mathsf{bp}, \mathsf{srs}, C, z^{-1}, c(X))$

$(k, W_k) \leftarrow \mathsf{PC.O}(\mathsf{bp}, \mathsf{srs}, C, y, c(X))$

$(t, W_t) \leftarrow \mathsf{PC.O}(\mathsf{bp}, \mathsf{srs}, T, z, t(X, y))$ $\xrightarrow{a, W_a, b, W_b, k, W_k, W_t}$

$t \leftarrow (ya + z^{n+2} + z^{n+1} y$

$-z^{2n+2} y)(c + 1) z^{-1} - ky - 1$

If $e(g^{\alpha a_{n+1} x^{n+1}} U, h) e(V, h^{x^{n+1}})$

$\neq e(A, h)$

$\vee \mathsf{PC.V}(\mathsf{bp}, \mathsf{srs}, d, A, yz, (a, W_a)) = 0$

$\vee \mathsf{PC.V}(\mathsf{bp}, \mathsf{srs}, d, C, z^{-1}, (c, W_c)) = 0$

$\vee \mathsf{PC.V}(\mathsf{bp}, \mathsf{srs}, d, C, y, (k, W_k)) = 0$

$\vee \mathsf{PC.V}(\mathsf{bp}, \mathsf{srs}, d, T, z, (t, W_t)) = 0$

$\vee \mathsf{SnWF.V}(\mathsf{bp}, \mathsf{srs}, 2n + 1, C, C_w) = 0$

$\vee \mathsf{SnWF.V}(\mathsf{bp}, \mathsf{srs}, n, U, U_w) = 0$

$\vee \mathsf{SnWF.V}(\mathsf{bp}, \mathsf{srs}, n, V, V_w) = 0$

  Return 0

Return 1

Fig. 20. The grand product argument in Sonic.

$\underline{\mathsf{SnWF.Prv}(\mathsf{bp}, \mathsf{srs}, n, f(X))}$
$W \leftarrow (g^{x^{-d} f(x)}, g^{x^{d-n} f(x)})$
Return W

$\underline{\mathsf{SnWF.V}(\mathsf{bp}, \mathsf{srs}, n, F, W = (L, R))}$
If $e(F, h) = e(L, h^{\alpha x^d}) \wedge e(F, h) = e(R, h^{\alpha x^{n-d}})$ then
  Return 1
Return 0

Fig. 21. Well-formedness argument in Sonic

Proof. (Theorem 7) We shall invoke Theorem 1 by defining BadCh and e and showing that $\varepsilon \leqslant \frac{18n}{p-1}$ and there exists adversaries $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$ such that

$$p_{\mathsf{fail}}(\mathsf{SnACSPf}, \mathcal{P}_{\mathsf{alg}}, \mathsf{e}, R, \lambda) \leqslant \mathsf{Adv}^{4n\text{-}\mathsf{dl}}_{\mathbb{G}}(\mathcal{F}_1) + \mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{F}_2) + \mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{F}_3) \ .$$

DEFINING BadCh AND UPPER BOUNDING ε. To start off, we shall define $\mathsf{BadCh}(\tau')$ for all partial extended transcripts τ'. Let Ch be the set from which the challenge that comes right after τ' is sampled. We define a helper function CheckBad that takes as input a partial extended transcripts $[\tau']$ and a challenge $c \in \mathsf{Ch}$ and returns `true` if and only if $c \in \mathsf{BadCh}(\tau')$. Since SnACSPf has two challenges, there are two definitions of CheckBad in Figure 22. We again use the predicate SZ here like before. Next, we need to compute an upper bound ε on the size of $|\mathsf{BadCh}(\tau')|/|\mathsf{Ch}|$. In other words, we need to compute an upper bound on the fraction of c's in Ch that $\mathsf{CheckBad}(\tau', c)$ will return `true` for all the definitions of CheckBad.

The function $\mathsf{CheckBad}(\tau', y)$ returns `true` if $\mathsf{SZ}(f(Y), y)$ is `true`. We shall use the Schwartz-Zippel lemma to fraction bound the number of y's that $\mathsf{SZ}(f(Y), y)$ is true for $y \in \mathbb{Z}_p^*$. The polynomial $f(Y)$ is a polynomial of degree at most $2n + Q$ (the maximum positive degree is $n + Q$ while the maximum negative degree is $-n$). Since $Q \leqslant 2n$, the degree of $f(Y)$ is at most $4n$. So, for at most at most $4n$ values of $y \in \mathbb{Z}_p^*$, $\mathsf{SZ}(f(Y), y)$ is `true`. So the $\mathsf{CheckBad}(\tau', y)$ returns `true` for at most $4n/(p-1)$ fraction of y's.

The function $\mathsf{CheckBad}(\tau', z)$ returns `true` if $\mathsf{SZ}(f(Z), z)$ is `true`. The polynomial $f(Z)$ is a polynomial of degree at most $18n$ (the maximum positive degree is $d < 4n$ while the maximum negative degree is $2n - 4d > -16d$). So, the fraction of z's in $\mathbb{Z}_p^*$ for which $\mathsf{SZ}(f(Z), z)$ is `true` is at most $18n/(p-1)$. So the fraction of z's in $\mathbb{Z}_p^*$ for which $\mathsf{CheckBad}(\tau', z)$ returns `true` is at most $18n/(p-1)$.

Similarly we can argue that for $j = 1, \ldots, M$, $\mathsf{CheckBad}(\tau', z_j)$ returns `true` with probability at most $(10n+1)/(p-1)$, $\mathsf{CheckBad}(\tau', y_j)$ returns `true` with probability at most $(2n+2)/(p-1)$, $\mathsf{CheckBad}(\tau', (\beta_j, \gamma_j))$ returns `true` with probability at most $n/(p-1)$, $\mathsf{CheckBad}(\tau', \delta_j)$ returns `true` with probability at most $8n/(p-1)$.

Therefore $\mathsf{CheckBad}(\tau', c)$ will return `true` for any partial transcript τ' for a no more than $18n/(p-1)$ values of c, i.e., in the context of the Master Theorem $\varepsilon \leqslant \frac{18n}{p-1}$.

DEFINING e. Next, we define the function e for SnACSPf in Figure 23. It gets as input an extended accepting transcript $[\tau]$ with the representation of the input removed. Without loss of generality we assume that the representations of all the messages of the prover in the transcript that are from $\mathbb{G}$ are in terms of the elements of $\mathbb{G}$ in srs. The function e computes $(\mathbf{a}_L^*, \mathbf{a}_R^*, \mathbf{a}_O^*)$ and outputs them. It follows from the description of e that it runs in time $O(n)$. Note that SnACSPf.V runs in time $O(n)$. Therefore, using Theorem 1, the time complexity of $\mathcal{E}$ is $O(q \cdot n)$.

PROVING AN UPPER BOUND ON $p_{\mathsf{fail}}(\mathsf{SnACSPf}, \mathcal{P}_{\mathsf{alg}}, \mathsf{e}, R, \lambda)$. To that end, we construct the following three adversaries $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$.

Procedure CheckBad($[\tau']$, z):

$f(Z) \leftarrow \left(\sum_{\substack{i=-d \\ i\neq 0}}^{d} Z^i t_{g^{\alpha x^i}}\right) - \left(\sum_{\substack{i=n-2d \\ i\neq n-d}}^{n} Z^i r_{g^{\alpha x^{i-n+d}}}\right)\left(\sum_{\substack{i=n-2d \\ i\neq n-d}}^{n} (yZ)^i r_{g^{\alpha x^{i-n+d}}} - s(Z,y)\right) + k(y)$

Return $\mathsf{SZ}(f(Z), z)$

Procedure CheckBad($[\tau']$, y):

For $i = 1, \ldots, n$ do

$a_i^* \leftarrow r_{g^{\alpha x^{d-n+i}}}$; $b_i^* \leftarrow r_{g^{\alpha x^{d-n-i}}}$; $c_i^* \leftarrow r_{g^{\alpha x^{d-2n-i}}}$

$\mathbf{a}_L^* \leftarrow (a_1^*, \ldots, a_n^*)$; $\mathbf{a}_R^* \leftarrow (b_1^*, \ldots, b_n^*)$; $\mathbf{a}_O^* \leftarrow (c_1^*, \ldots, c_n^*)$

$f(Y) \leftarrow r_{g^{\alpha x^{d-n}}} r_{g^{\alpha x^{d-n}}} + \langle \mathbf{a}_L^* \circ \mathbf{a}_R^* - \mathbf{a}_O^*, \mathbf{Y}_{[1:]}^{n+1} + \mathbf{Y}_{[1:]}^{-n-1} \rangle + \mathbf{Y}_{[n+1:]}^{Q+n+1} \cdot (\mathbf{W}_L \cdot \mathbf{a}_L^* + \mathbf{W}_R \cdot \mathbf{a}_R^* + \mathbf{W}_O \cdot \mathbf{a}_O^*) - \langle \mathbf{c}, \mathbf{Y}_{[n+1:]}^{Q+n+1} \rangle$

Return $\mathsf{SZ}(f(Y), y)$

Procedure CheckBad($[\tau']$, z_j):

$f(Z) \leftarrow \left(\sum_{\substack{i=-d \\ i\neq 0}}^{d} Z^i t_{jg^{\alpha x^i}}\right) - \left(y_j \left(\sum_{\substack{i=-d \\ i\neq 0}}^{d} (y_j Z)^i a_{jg^{\alpha x^i}}\right) + Z^{n+2} + Z^{n+1} y - Z^{2n+2} y_j\right) \cdot \left(\sum_{i=1}^{2n+1} Z^{-i} c_{jg^{\alpha x^i}} + 1\right) Z^{-1} + \left(\sum_{i=1}^{2n+1} y_j^i c_{jg^{\alpha x^i}}\right) y_j + 1$

Return $\mathsf{SZ}(f(Z), z_j)$

Procedure CheckBad($[\tau']$, y_j):

$f(Y) \leftarrow \sum_{i=1}^{2n+1} Y^{i+1} \left(a_{jg^{\alpha x^i}} c_{jg^{\alpha x^{i-1}}} - c_{jg^{\alpha x^{i-1}}}\right) + (c_{jg^{\alpha x^{n+1}}} - 1) + Y(c_{jg^{\alpha x^n}} - c_{jg^{\alpha x^{2n+1}}})$

Return $\mathsf{SZ}(f(Y), y_j)$

Procedure CheckBad($[\tau']$, (β_j, γ_j)):

$f(\mathrm{B}, \Gamma) \leftarrow \prod_{i=1}^{n} \left(s_{jg^{\alpha x^i}} + \mathrm{B}\sigma_{j,i} + \Gamma\right) - \prod_{i=1}^{n} \left(s'_{jg^{\alpha x^i}} + \mathrm{B}i + \Gamma\right)$

Return $\mathsf{SZ}(f(\mathrm{B}, \Gamma), (\beta_j, \gamma_j))$

Procedure CheckBad($[\tau']$, δ_j):

$f(\Delta) \leftarrow \left(\sum_{\substack{i=-d \\ i\neq 0}}^{d} \Delta^i s'_{g^{\alpha x^i}}\right) - \left(\sum_{\substack{i=-d \\ i\neq 0}}^{d} (\Delta y_j)^i \psi_i\right)$

Return $\mathsf{SZ}(f(\Delta), \delta_j)$

Fig. 22. The function CheckBad function for the SnACSPf.

Procedure e($[\tau]$):

$//[\tau] = \Big(\mathsf{bp}, \mathsf{srs}, n, d, s(X,Y), k(Y), \{\psi_j(X,Y), \sigma_j\}_{j=1}^{M}; [R], y, [T], z, (a, [W_a], b, [W_b], [W_t], s), \{([S_i, S_i']), (\delta_i, \beta_i, \gamma_i),$

$(s_i, [W_i], v_i, [W_i'], [Q_i']), ([A_i, C_i, C_{i,w}, U_{i,w}, V_{i,w}], a_{i,n+1}), y_i, T_i, z_i, (a_i, [W_{i,a}], b_i, [W_{i,b}], k_i, [W_{i,k}], [W_{i,t}])\}_{i=1}^{M}\Big)$

$a_i^* \leftarrow r_{g^{\alpha x^{d-n+i}}}$; $b_i^* \leftarrow r_{g^{\alpha x^{d-n-i}}}$; $c_i^* \leftarrow r_{g^{\alpha x^{d-2n-i}}}$

$\mathbf{a}_L^* \leftarrow (a_1^*, \ldots, a_n^*)$; $\mathbf{a}_R^* \leftarrow (b_1^*, \ldots, b_n^*)$; $\mathbf{a}_O^* \leftarrow (c_1^*, \ldots, c_n^*)$

Return $(\mathbf{a}_L^*, \mathbf{a}_R^*, \mathbf{a}_O^*)$

Fig. 23. The function e for SnACSPf.

1. Adversary $\mathcal{F}_1$ is an adversary against d-DLOG in the group $\mathbb{G}$ that runs $\mathcal{P}_{\mathsf{alg}}$. It has inputs $(g, g^x, g^{x^2}, \ldots, g^{x^d})$. It fixes a positive integer n such that $4n > d > 3n$. It samples $\alpha, \beta \in \mathbb{Z}_p$, and

Procedure $\mathsf{h}_1([\tau], \alpha, \beta, X)$:

$//[\tau] = \Big(\mathsf{bp}, \mathsf{srs}, n, d, s(X,Y), k(Y), \{\psi_j(X,Y), \sigma_j\}_{j=1}^{M}; [R], y, [T], z, (a, [W_a], b, [W_b], [W_t], s), \{([S_i, S_i']), (\delta_i, \beta_i, \gamma_i),$

$(s_i, [W_i], v_i, [W_i'], [Q_i']), ([A_i, C_i, C_{i,w}, U_{i,w}, V_{i,w}], a_{i,n+1}), y_i, T_i, z_i, (a_i, [W_{i,a}], b_i, [W_{i,b}], k_i, [W_{i,k}], [W_{i,t}])\}_{i=1}^{M}\Big)$

$x^* \leftarrow \mathsf{ComHelper}_1(X, [R], [W_a], a, z, n)$
If $x^* \neq \perp$ then return x^*
$x^* \leftarrow \mathsf{ComHelper}_1(X, [R], [W_b], b, yz, n)$
If $x^* \neq \perp$ then return x^*
$t \leftarrow a(b+s) - k(y)$
$x^* \leftarrow \mathsf{ComHelper}_1(X, [T], [W_t], t, z, d)$
If $x^* \neq \perp$ then return x^*
For $i = 1, \ldots, M$ do
  $x^* \leftarrow \mathsf{ComHelper}_1(X, [S_i], [W_i], s_i, z, d)x$; If $x^* \neq \perp$ then return x^*
  $x^* \leftarrow \mathsf{ComHelper}_1(X, [S_i'], [Q_i], v_i, \delta_i, d)$; If $x^* \neq \perp$ then return x^*
  $x^* \leftarrow \mathsf{ComHelper}_1(X, [P_{2,i}], [Q_i'], v_i, \delta_i y, d)$; If $x^* \neq \perp$ then return x^*
  $x^* \leftarrow \mathsf{ComHelper}_1(X, [C_i], [W_{i,c}], c_i, z^{-1}, d)$; If $x^* \neq \perp$ then return x^*
  $x^* \leftarrow \mathsf{ComHelper}_1(X, [C_i], [W_{i,k}], k_i, y, d)$; If $x^* \neq \perp$ then return x^*
  $x^* \leftarrow \mathsf{ComHelper}_1(X, [T_i], [W_{i,t}], t_i, z, d)$; If $x^* \neq \perp$ then return x^*
  $x^* \leftarrow \mathsf{EqHelper}_1(X, [U_i], [V_i], [A_i], a_{i,n+1}, n)$; If $x^* \neq \perp$ then return x^*
  $x^* \leftarrow \mathsf{WfHelper}_1(X, [C], [C_{i,w,1}, C_{i,w,2}], 2n+1)$;If $x^* \neq \perp$ then return x^*
  $x^* \leftarrow \mathsf{WfHelper}_1(X, [U], [U_{i,w,1}, U_{i,w,2}], n)$;If $x^* \neq \perp$ then return x^*
  $x^* \leftarrow \mathsf{WfHelper}_1(X, [V], [V_{i,w,1}, V_{i,w,2}], n)$
  If $x^* \neq \perp$ then return x^*
Return $\perp$

Fig. 24. The function h_1 for SnACSPf.

sets $\mathsf{bp} = (p, \mathbb{G}, \mathbb{G}_T, e, g, g^\beta)$ and

$$\mathsf{srs} = \{g, \{g^{x^i}\}_{i=-d}^{d}, \{g^{x^i\beta}\}_{i=-d}^{d}, \{g^{x^i\alpha\beta}\}_{i=-d}^{d}, \{g^{x^i\alpha}\}_{\substack{i=-d \\ i\neq 0}}^{d}, e(g, g^{\alpha\beta})\} .$$

Note that $(n, d, \mathsf{bp}, \mathsf{srs})$ is a valid output of SnACSPf.Setup. Adversary $\mathcal{F}_1$ runs $\mathcal{P}_{\mathsf{alg}}$ on public parameters $(n, d, \mathsf{bp}, \mathsf{srs})$ and simulates the game $\mathsf{SRS}_{\mathsf{SnACSPf}}$ to it. If $\mathcal{P}_{\mathsf{alg}}$ manages to produce an accepting transcript τ, $\mathcal{F}_1$ calls a helper function h_1 on input $[\tau], \alpha, \beta, g^x$ and outputs whatever h_1 outputs. The function h_1 is defined in Figure 24. The subroutines used in h_1 are defined in Figures 27 to 29.

2. Adversary $\mathcal{F}_2$ is an adversary against DLOG in the group $\mathbb{G}$ that runs $\mathcal{P}_{\mathsf{alg}}$. It has inputs (g, V). It fixes a positive integer n such that $4n > d > 3n$. It samples $\alpha, x \in \mathbb{Z}_p$, and sets $\mathsf{bp} = (p, \mathbb{G}, \mathbb{G}_T, e, g, V)$ and

$$\mathsf{srs} = \{g, \{g^{x^i}\}_{i=-d}^{d}, \{V^{x^i}\}_{i=-d}^{d}, \{V^{x^i\alpha}\}_{i=-d}^{d}, \{g^{x^i\alpha}\}_{\substack{i=-d \\ i\neq 0}}^{d}, e(g, V^\alpha)\} .$$

Note that $(n, d, \mathsf{bp}, \mathsf{srs})$ is a valid output of SnACSPf.Setup. Adversary $\mathcal{F}_2$ runs $\mathcal{P}_{\mathsf{alg}}$ on public parameters $(n, d, \mathsf{bp}, \mathsf{srs})$ and simulates the game $\mathsf{SRS}_{\mathsf{SnACSPf}}$ to it. If $\mathcal{P}_{\mathsf{alg}}$ manages to produce an accepting transcript τ, $\mathcal{F}_2$ calls a helper function h_2 on input $[\tau], x, \alpha, V$ and outputs whatever h_2 outputs. The function h_2 is defined in Figure 25. The subroutines used in h_2 are defined in Figures 27 to 29.

Procedure $\mathsf{h}_2([\tau], \alpha, x, X)$:

$//[\tau] = \Big(\mathsf{bp}, \mathsf{srs}, n, d, s(X,Y), k(Y), \{\psi_j(X,Y), \sigma_j\}_{j=1}^{M}; [R], y, [T], z, (a, [W_a], b, [W_b], [W_t], s), \{([S_i, S_i']), (\delta_i, \beta_i, \gamma_i),$
$(s_i, [W_i], v_i, [W_i'], [Q_i']), ([A_i, C_i, C_{i,w}, U_{i,w}, V_{i,w}], a_{i,n+1}), y_i, T_i, z_i, (a_i, [W_{i,a}], b_i, [W_{i,b}], k_i, [W_{i,k}], [W_{i,t}])\}_{i=1}^{M}\Big)$
$\beta^* \leftarrow \mathsf{ComHelper}_2(X, x, \alpha, [R], [W_a], a, z, n)$
If $\beta^* \neq \perp$ then return β^*
$\beta^* \leftarrow \mathsf{ComHelper}_2(X, x, \alpha, [R], [W_b], b, yz, n)$
If $\beta^* \neq \perp$ then return β^*
$t \leftarrow a(b+s) - k(y)$
$\beta^* \leftarrow \mathsf{ComHelper}_2(X, x, \alpha, [T], [W_t], t, z, d)$
If $\beta^* \neq \perp$ then return β^*
For $i = 1, \ldots, M$ do
  $\beta^* \leftarrow \mathsf{ComHelper}_2(X, x, \alpha, [S_i], [W_i], s_i, z, d)x$; If $\beta^* \neq \perp$ then return β^*
  $\beta^* \leftarrow \mathsf{ComHelper}_2(X, x, \alpha, [S_i'], [Q_i], v_i, \delta_i, d)$; If $\beta^* \neq \perp$ then return β^*
  $\beta^* \leftarrow \mathsf{ComHelper}_2(X, x, \alpha, [P_{2,i}], [Q_i'], v_i, \delta_i y, d)$; If $\beta^* \neq \perp$ then return β^*
  $\beta^* \leftarrow \mathsf{ComHelper}_2(X, x, \alpha, [C_i], [W_{i,c}], c_i, z^{-1}, d)$; If $\beta^* \neq \perp$ then return β^*
  $\beta^* \leftarrow \mathsf{ComHelper}_2(X, x, \alpha, [C_i], [W_{i,k}], k_i, y, d)$; If $\beta^* \neq \perp$ then return β^*
  $\beta^* \leftarrow \mathsf{ComHelper}_2(X, x, \alpha, [T_i], [W_{i,t}], t_i, z, d)$; If $\beta^* \neq \perp$ then return β^*
  $\beta^* \leftarrow \mathsf{EqHelper}_2(X, x, \alpha, [U_i], [V_i], [A_i], a_{i,n+1}, n)$; If $\beta^* \neq \perp$ then return β^*
  $\beta^* \leftarrow \mathsf{WfHelper}_2(X, x, \alpha, [C], [C_{i,w,1}, C_{i,w,2}], 2n+1)$;If $\beta^* \neq \perp$ then return β^*
  $\beta^* \leftarrow \mathsf{WfHelper}_2(X, x, \alpha, [U], [U_{i,w,1}, U_{i,w,2}], n)$;If $\beta^* \neq \perp$ then return β^*
  $\beta^* \leftarrow \mathsf{WfHelper}_2(X, x, \alpha, [V], [V_{i,w,1}, V_{i,w,2}], n)$; If $\beta^* \neq \perp$ then return β^*
Return $\perp$

Fig. 25. The function h_2 for SnACSPf.

3. Adversary $\mathcal{F}_3$ is an adversary against DLOG in the group $\mathbb{G}$ that runs $\mathcal{P}_{\mathsf{alg}}$. It has inputs (g, V). It fixes a positive integer n such that $4n > d > 3n$. It samples $\beta, x \in \mathbb{Z}_p$, and sets $\mathsf{bp} = (p, \mathbb{G}, \mathbb{G}_T, e, g, g^\beta)$ and

$$\mathsf{srs} = \{g, \{g^{x^i}\}_{i=-d}^{d}, \{g^{\beta x^i}\}_{i=-d}^{d}, \{V^{x^i \beta}\}_{i=-d}^{d}, \{V^{x^i}\}_{\substack{i=-d \\ i\neq 0}}^{d}, e(g, V^\beta)\} .$$

   Note that $(n, d, \mathsf{bp}, \mathsf{srs})$ is a valid output of SnACSPf.Setup. Adversary $\mathcal{F}_3$ runs $\mathcal{P}_{\mathsf{alg}}$ on public parameters $(n, d, \mathsf{bp}, \mathsf{srs})$ and simulates the game $\mathsf{SRS}_{\mathsf{SnACSPf}}$ to it. If $\mathcal{P}_{\mathsf{alg}}$ manages to produce an accepting transcript τ, $\mathcal{F}_3$ calls a helper function h_3 on input $[\tau], x, \beta, V$ and outputs whatever h_3 outputs. The function h_3 is defined in Figure 26. The subroutines used in h_3 are defined in Figures 27 to 29.

Note that the definitions of the helper functions are modular, i.e., h_i use the subroutines PC_i, $\mathsf{EqHelper}_i$, $\mathsf{WfHelper}_i$ for $i = 1, 2, 3$. The subroutine PC_i produces a solution to the relevant hard problem if the prover manages to break the binding of any of the commitment. Similarly the subroutine $\mathsf{WfHelper}_i$ produces a solution to the relevant hard problem if the prover manages to pass the well-formedness verification for a commitment that is not well-formed.

We now make the following observations about adversaries $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$

- Adversary $\mathcal{F}_1$ succeeds if $\mathsf{h}_1([\tau], \alpha, \beta)$ computes x^* such that $(g^{x^*} = g^x)$. From the code of h_1 it is easy to see that that whenever h_1 returns a non-$\perp$ value x^*, it satisfies $(g^{x^*} = g^x)$, i.e., adversary $\mathcal{F}_1$ succeeds. Also, it follows from the description of h_1 that it runs in time at $O(n)$

Procedure $\mathsf{h}_3([\tau], x, \beta, X)$:

$//[\tau] = \Big(\mathsf{bp}, \mathsf{srs}, n, d, s(X,Y), k(Y), \{\psi_j(X,Y), \sigma_j\}_{j=1}^{M}; [R], y, [T], z, (a, [W_a], b, [W_b], [W_t], s), \{(\big[S_i, S_i'\big]), (\delta_i, \beta_i, \gamma_i),$
$(s_i, [W_i], v_i, \big[W_i'\big], \big[Q_i'\big]), ([A_i, C_i, C_{i,w}, U_{i,w}, V_{i,w}], a_{i,n+1}), y_i, T_i, z_i, (a_i, [W_{i,a}], b_i, [W_{i,b}], k_i, [W_{i,k}], [W_{i,t}])\}_{i=1}^{M}\Big)$
$\alpha^* \leftarrow \mathsf{ComHelper}_3(X, x, [R], [W_a], a, z, n)$
If $\alpha^* \neq \perp$ then return α^*
$\alpha^* \leftarrow \mathsf{ComHelper}_3(X, x, [R], [W_b], b, yz, n)$
If $\alpha^* \neq \perp$ then return α^*
$t \leftarrow a(b+s) - k(y)$
$\alpha^* \leftarrow \mathsf{ComHelper}_3(X, x, [T], [W_t], t, z, d)$
If $\alpha^* \neq \perp$ then return α^*
For $i = 1, \ldots, M$ do
  $\alpha^* \leftarrow \mathsf{ComHelper}_3(X, a, x, [S_i], [W_i], s_i, z, d)x$; If $\alpha^* \neq \perp$ then return α^*
  $\alpha^* \leftarrow \mathsf{ComHelper}_3(X, x, \big[S_i'\big], [Q_i], v_i, \delta_i, d)$; If $\alpha^* \neq \perp$ then return α^*
  $\alpha^* \leftarrow \mathsf{ComHelper}_3(X, x, \big[P_{2,i}\big], \big[Q_i'\big], v_i, \delta_i y, d)$; If $\alpha^* \neq \perp$ then return α^*
  $\alpha^* \leftarrow \mathsf{ComHelper}_3(X, x, [C_i], [W_{i,c}], c_i, z^{-1}, d)$; If $\alpha^* \neq \perp$ then return α^*
  $\alpha^* \leftarrow \mathsf{ComHelper}_3(X, x, [C_i], [W_{i,k}], k_i, y, d)$; If $\alpha^* \neq \perp$ then return α^*
  $\alpha^* \leftarrow \mathsf{ComHelper}_3(X, x, [T_i], [W_{i,t}], t_i, z, d)$; If $\alpha^* \neq \perp$ then return α^*
  $\alpha^* \leftarrow \mathsf{EqHelper}_3(X, x, [U_i], [V_i], [A_i], a_{i,n+1}, n)$; If $\alpha^* \neq \perp$ then return α^*
  $\alpha^* \leftarrow \mathsf{WfHelper}_3(X, x, [C], [C_{i,w,1}, C_{i,w,2}], 2n+1)$;If $\alpha^* \neq \perp$ then return α^*
  $\alpha^* \leftarrow \mathsf{WfHelper}_3(X, x, [U], [U_{i,w,1}, U_{i,w,2}], n)$;If $\alpha^* \neq \perp$ then return α^*
  $\alpha^* \leftarrow \mathsf{WfHelper}_3(X, x, [V], [V_{i,w,1}, V_{i,w,2}], n)$; If $\alpha^* \neq \perp$ then return α^*
Return $\perp$

Fig. 26. The function h_3 for SnACSPf.

(since $M = O(1)$). The running time of $\mathcal{F}_1$ consists of the time required to answers q queries, run SnACSPf.V in at most q paths in the execution tree and the time required to run h_1. Hence its time complexity is $O(qn)$.

- Adversary $\mathcal{F}_2$ succeeds if $\mathsf{h}_2([\tau], x, \alpha)$ computes β^* such that $g^{\beta^*} = V$. From the code of h_2 it is easy to see that that whenever h_2 returns a non-$\perp$ value β^*, it satisfies $(g^{\beta^*} = h)$, i.e., adversary $\mathcal{F}_2$ succeeds. Also, it follows from the description of h_2 that it runs in time $O(n)$ (since $M = O(1)$). The running time of $\mathcal{F}_2$ consists of the time required to answers q queries, run SnACSPf.V in at most q paths in the execution tree and the time required to run h_2. Hence its time complexity is $O(qn)$.
- Adversary $\mathcal{F}_3$ succeeds if $\mathsf{h}_3(\tau, x, \beta)$ computes α^* such that $g^{\alpha^*} = V$. From the code of h_3 it is easy to see that that whenever h_3 returns a non-$\perp$ value α^*, it satisfies $(g^{\alpha^*} = g^x)$, i.e., adversary $\mathcal{F}_3$ succeeds. Also, it follows from the description of h_3 that it runs in time $O(n)$ (since $M = O(1)$). The running time of $\mathcal{F}_3$ consists of the time required to answers q queries, run SnACSPf.V in at most q paths in the execution tree and the time required to run h_3. Hence its time complexity is $O(qn)$.

We shall prove the following lemma showing that if τ is an accepting transcript such that $\tau \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{SnACSPf}}$ and $\mathsf{h}_1([\tau], \alpha, \beta)$, $\mathsf{h}_2([\tau], x, \alpha)$, $\mathsf{h}_3(\tau, x, \beta)$ all return $\perp$, then $\mathsf{e}([\tau])$ returns a valid witness.

Procedure $\mathsf{ComHelper}_1(X, [C], [W], v, z, n)$

$f(X) \leftarrow (X - z)\left(\sum_{i=-d}^{d} X^i w_{g^{x^i}}\right) + v - X^{-d+n}\left(\sum_{\substack{i=-d \\ i\neq 0}}^{d} X^i c_{g^{\alpha x^i}}\right)$

If $f(X) \neq 0$ then
  Solve for x^* such that $f(x^*) = 0$
  If $X = g^{x^*}$ then return x^*
Return $\perp$

Procedure $\mathsf{ComHelper}_2(X, x, \alpha, [C], [W], v, z, n)$:

$$f(\mathrm{B}) \leftarrow \mathrm{B}\left(\alpha(x-z)\left(\sum_{i=-d}^{d} x^i w_{h^{x^i}} + \alpha x^i w_{h^{\alpha x^i}}\right) - x^{-d+n}\left(\sum_{i=-d}^{d} x^i c_{h^{x^i}} + \alpha x^i c_{h^{\alpha x^i}}\right)\right)$$
$$+ \alpha(x-z)\left(\sum_{i=-d}^{d} x^i w_{g^{x^i}} + \sum_{\substack{i=-d \\ i\neq 0}}^{d} \alpha x^i w_{g^{\alpha x^i}}\right) + \alpha v - x^{-d+n}\left(\sum_{i=-d}^{d} x^i c_{g^{x^i}} + \sum_{\substack{i=-d \\ i\neq 0}}^{d} \alpha x^i c_{g^{\alpha x^i}}\right)$$

If $f(\mathrm{B}) \neq 0$ then
  Solve for β^* such that $f(\beta^*) = 0$
  If $X = g^{\beta^*}$ then return β^*
Return $\perp$

Procedure $\mathsf{ComHelper}_3(X, x, [C], [W], v, z, n)$

$$f(\mathrm{A}) \leftarrow \mathrm{A}(x-z)\left(\sum_{i=-d}^{d} x^i w_{g^{x^i}} + \sum_{\substack{i=-d \\ i\neq 0}}^{d} \mathrm{A} x^i w_{g^{\alpha x^i}}\right) + \mathrm{A}v - x^{-d+n}\left(\sum_{i=-d}^{d} x^i c_{g^{x^i}} + \sum_{\substack{i=-d \\ i\neq 0}}^{d} \mathrm{A} x^i c_{g^{\alpha x^i}}\right)$$

If $f(\mathrm{A}) \neq 0$ then
  Solve for α^* such that $f(\alpha^*) = 0$
  If $X = g^{\alpha^*}$ then return α^*
Return $\perp$

Fig. 27. Subroutines for $\mathsf{h}_1, \mathsf{h}_2, \mathsf{h}_3$.

Lemma 11. *Let $n, d \in \mathbb{N}$ such that $d > 3n$. Let $x, \alpha, \beta \in \mathbb{Z}_p$, $\mathsf{bp} = (p, \mathbb{G}, \mathbb{G}_T, e, g, h)$. Let $\mathsf{srs} = \{g, \{g^{x^i}\}_{i=-d}^{d}, \{g^{\beta x^i}\}_{i=-d}^{d}, \{g^{\alpha\beta x^i}\}_{i=-d}^{d}, \{g^{\alpha x^i}\}_{\substack{i=-d \\ i\neq 0}}^{d}, e(g, h^\alpha)\}$. Let*

$$\tau = \big(\mathsf{bp}, \mathsf{srs}, n, d, s(X,Y), k(Y), \{\psi_j(X,Y), \sigma_j\}_{j=1}^{M}; [R], y, [T], z, (a, [W_a], b, [W_b],$$
$$[W_t], s), \{([S_i, S_i']), (\delta_i, \beta_i, \gamma_i), (s_i, [W_i], v_i, [W_i'], [Q_i']), ([A_i, C_i, C_{i,w}, U_{i,w}, V_{i,w}],$$
$$a_{i,n+1}), y_i, T_i, z_i, (a_i, [W_{i,a}], b_i, [W_{i,b}], k_i, [W_{i,k}], [W_{i,t}])\}_{i=1}^{M}$$

be an accepting transcript of $\mathsf{SnACSPf}$ *such that $\tau \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{SnACSPf}}$. If $\mathsf{h}_1([\tau], \alpha, \beta, g^x)$, $\mathsf{h}_2([\tau], x, \alpha, V)$ and $\mathsf{h}_3([\tau], x, \beta, V)$ return $\perp$, then $\mathsf{e}([\tau])$ returns $(\mathbf{a}_L^*, \mathbf{a}_R^*, \mathbf{a}_O^*)$ such that*

$$\mathbf{a}_L^* \circ \mathbf{a}_R^* = \mathbf{a}_O^* \quad \textit{and} \quad \mathbf{a}_L^* \cdot \mathbf{W}_L + \mathbf{a}_R^* \cdot \mathbf{W}_R + \mathbf{a}_O^* \cdot \mathbf{W}_O = \mathbf{c}\ .$$

Taking the contrapositive, we get that if τ is an accepting transcript such that $\tau \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{SnACSPf}}$ and $\mathsf{e}([\tau])$ fails to return a valid witness, then one of $\mathsf{h}_1([\tau], \alpha, \beta, g^x)$, $\mathsf{h}_2([\tau], x, \alpha, V)$, $\mathsf{h}_3(\tau, x, \beta, V)$ returns a non-$\perp$ value, i.e., one of adversaries $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$ succeed. Therefore

$$p_{\mathsf{fail}}(\mathsf{SnACSPf}, \mathcal{P}_{\mathsf{alg}}, \mathsf{e}, R, \lambda) \leqslant \mathsf{Adv}_{\mathbb{G}}^{4n\text{-}\mathsf{dl}}(\mathcal{F}_1) + \mathsf{Adv}_{\mathbb{G}}^{\mathsf{dl}}(\mathcal{F}_2) + \mathsf{Adv}_{\mathbb{G}}^{\mathsf{dl}}(\mathcal{F}_3)\ .$$

□

We shall next prove Lemma 11.

Procedure $\mathsf{EqHelper}_1(X, [U], [V], [A], a_{n+1}, n)$

$$f(X) \leftarrow \left(\sum_{\substack{i=-d \\ i\neq 0}}^{d} X^i u_{g^{\alpha x^i}} + X^{i+n+1} v_{g^{\alpha x^i}} \right) + a_{n+1} X^{n+1} - \left(\sum_{\substack{i=-d \\ i\neq 0}}^{d} X^i a_{g^{\alpha x^i}} \right)$$

If $f(X) \neq 0$ then
  Solve for x^* such that $f(x^*) = 0$
  If $X = g^{x^*}$ then return x^*
Return $\perp$

Procedure $\mathsf{EqHelper}_2(X, x, \alpha, [U], [V], [A], a_{n+1}, n)$:

$$\begin{aligned} f(\mathrm{B}) \leftarrow & \left(\sum_{i=-d}^{d} x^i u_{g^{x^i}} + x^{i+n+1} v_{g^{x^i}} + \sum_{\substack{i=-d \\ i\neq 0}}^{d} \alpha x^i u_{g^{\alpha x^i}} + x^{i+n+1} v_{g^{\alpha x^i}} \right) \\ & + \mathrm{B} \left(\sum_{i=-d}^{d} x^i u_{h^{x^i}} + \alpha x^i u_{h^{\alpha x^i}} + x^{i+n+1} v_{h^{x^i}} + \alpha x^{i+n+1} v_{h^{\alpha x^i}} - \sum_{i=-d}^{d} x^i a_{h^{x^i}} + \alpha x^i a_{h^{\alpha x^i}} \right) + \alpha a_{n+1} x^{n+1} \\ & - \left(\sum_{i=-d}^{d} x^i a_{g^{x^i}} + \sum_{\substack{i=-d \\ i\neq 0}}^{d} \alpha x^i a_{g^{\alpha x^i}} \right) \end{aligned}$$

If $f(\mathrm{B}) \neq 0$ then
  Solve for β^* such that $f(\beta^*) = 0$
  If $X = g^{\beta^*}$ then return β^*
Return $\perp$

Procedure $\mathsf{EqHelper}_3(X, x, [U], [V], [A], a_{n+1}, n)$

$$f(\mathrm{A}) \leftarrow \left(\sum_{i=-d}^{d} x^i u_{g^{x^i}} + x^{i+n+1} v_{g^{x^i}} + \sum_{\substack{i=-d \\ i\neq 0}}^{d} \mathrm{A} x^i u_{g^{\alpha x^i}} + x^{i+n+1} v_{g^{\alpha x^i}} \right) + \mathrm{A} a_{n+1} x^{n+1} - \left(\sum_{i=-d}^{d} x^i a_{g^{x^i}} + \sum_{\substack{i=-d \\ i\neq 0}}^{d} \mathrm{A} x^i a_{g^{\alpha x^i}} \right)$$

If $f(\mathrm{A}) \neq 0$ then
  Solve for α^* such that $f(\alpha^*) = 0$
  If $X = g^{\alpha^*}$ then return α^*
Return $\perp$

Fig. 28. Subroutines for $\mathsf{h}_1, \mathsf{h}_2, \mathsf{h}_3$.

Proof (Lemma 11). Since τ is an accepting transcript the following equality holds.

$$e(W_a, h^{\alpha x}) e(g^a W_a^z, h^\alpha) = e(R, h^{x^{-d+n}}) \ .$$

We can express W_a in terms of its representations, let $h = g^\beta$ and re-write the first equality as

$$e(g, h)^f = 1 \ ,$$

where

$$\begin{aligned} f = & \alpha(x - z) \left(\sum_{i=-d}^{d} x^i w_{a g^{x^i}} + \beta x^i w_{a h^{x^i}} + \alpha\beta x^i w_{a h^{\alpha x^i}} + \sum_{\substack{i=-d \\ i\neq 0}}^{d} \alpha x^i w_{a g^{\alpha x^i}} \right) + \alpha a \\ & - x^{-d+n} \left(\sum_{i=-d}^{d} x^i r_{g^{x^i}} + \beta x^i r_{h^{x^i}} + \alpha\beta x^i r_{h^{\alpha x^i}} + \sum_{\substack{i=-d \\ i\neq 0}}^{d} \alpha x^i r_{g^{\alpha x^i}} \right) . \end{aligned}$$

Procedure $\mathsf{WfHelper}_1(X, [A], [L, R], n)$:

$f_1(X) \leftarrow x^d \left(\sum_{i=-d}^{d} X^i l_{g^{x^i}} \right) - \left(\sum_{\substack{i=-d \\ i \neq 0}}^{d} \mathrm{A} X^i a_{g^{\alpha x^i}} \right)$

If $f_1(X) \neq 0$ then
  Solve for x^* such that $f_1(x^*) = 0$
  If $X = g^{x^*}$ then return x^*
Return $\perp$

$f_2(X) \leftarrow X^{n-d} \left(\sum_{i=-d}^{d} X^i r_{g^{x^i}} \right) - \left(\sum_{\substack{i=-d \\ i \neq 0}}^{d} X^i a_{g^{\alpha x^i}} \right)$

If $f_2(X) \neq 0$ then
  Solve for x^* such that $f_2(x^*) = 0$
  If $X = g^{x^*}$ then return x^*
Return $\perp$

Procedure $\mathsf{WfHelper}_2(X, x, \alpha, [A], [L, R], n)$:

$$f_1(\mathrm{B}) \leftarrow \mathrm{B} \left(\alpha x^d \left(\sum_{i=-d}^{d} x^i l_{h^{x^i}} + \alpha x^i l_{h^{\alpha x^i}} \right) - \left(\sum_{i=-d}^{d} x^i a_{h^{x^i}} + \alpha x^i a_{h^{\alpha x^i}} \right) \right) + \alpha x^d \left(\sum_{i=-d}^{d} x^i l_{g^{x^i}} + \sum_{\substack{i=-d \\ i \neq 0}}^{d} \alpha x^i l_{g^{\alpha x^i}} \right) - \left(\sum_{i=-d}^{d} x^i a_{g^{x^i}} + \sum_{\substack{i=-d \\ i \neq 0}}^{d} \alpha x^i a_{g^{\alpha x^i}} \right)$$

If $f_1(\mathrm{B}) \neq 0$ then
  Solve for β^* such that $f_1(\beta^*) = 0$
  If $X = g^{\beta^*}$ then return β^*

$$f_2(\mathrm{B}) \leftarrow \mathrm{B} \left(\alpha x^{n-d} \left(\sum_{i=-d}^{d} x^i r_{h^{x^i}} + \alpha x^i r_{h^{\alpha x^i}} \right) - \left(\sum_{i=-d}^{d} x^i a_{h^{x^i}} + \alpha x^i a_{h^{\alpha x^i}} \right) \right) + \alpha x^{n-d} \left(\sum_{i=-d}^{d} x^i r_{g^{x^i}} + \sum_{\substack{i=-d \\ i \neq 0}}^{d} \alpha x^i r_{g^{\alpha x^i}} \right) - \left(\sum_{i=-d}^{d} x^i a_{g^{x^i}} + \sum_{\substack{i=-d \\ i \neq 0}}^{d} \alpha x^i a_{g^{\alpha x^i}} \right)$$

If $f_2(\mathrm{B}) \neq 0$ then
  Solve for β^* such that $f_2(\beta^*) = 0$
  If $X = g^{\beta^*}$ then return β^*
Return $\perp$

Procedure $\mathsf{WfHelper}_3(X, x, [A], [L, R], n)$:

$$f_1(\mathrm{A}) \leftarrow \mathrm{A} x^d \left(\sum_{i=-d}^{d} x^i l_{g^{x^i}} + \sum_{\substack{i=-d \\ i \neq 0}}^{d} \mathrm{A} x^i l_{g^{\alpha x^i}} \right) - \left(\sum_{i=-d}^{d} x^i a_{g^{x^i}} + \sum_{\substack{i=-d \\ i \neq 0}}^{d} \mathrm{A} x^i a_{g^{\alpha x^i}} \right)$$

If $f_1(\mathrm{A}) \neq 0$ then
  Solve for α^* such that $f_1(\alpha^*) = 0$
  If $X = g^{\alpha^*}$ then return α^*

$$f_2(\mathrm{A}) \leftarrow \mathrm{A} x^{n-d} \left(\sum_{i=-d}^{d} x^i r_{g^{x^i}} + \sum_{\substack{i=-d \\ i \neq 0}}^{d} \mathrm{A} x^i r_{g^{\alpha x^i}} \right) - \left(\sum_{i=-d}^{d} x^i a_{g^{x^i}} + \sum_{\substack{i=-d \\ i \neq 0}}^{d} \mathrm{A} x^i a_{g^{\alpha x^i}} \right)$$

If $f_2(\mathrm{A}) \neq 0$ then
  Solve for α^* such that $f_2(\alpha^*) = 0$
  If $X = g^{\alpha^*}$ then return α^*
Return $\perp$

Fig. 29. Subroutines for $\mathsf{h}_1, \mathsf{h}_2, \mathsf{h}_3$.

We therefore have

$$\begin{aligned}\alpha(x-z)\left(\sum_{i=-d}^{d} x^i w_{ag^{x^i}} + \beta x^i w_{ah^{x^i}} + \alpha\beta x^i w_{ah^{\alpha x^i}} + \sum_{\substack{i=-d\\ i\neq 0}}^{d} \alpha x^i w_{ag^{\alpha x^i}}\right)\\ + \alpha a - x^{-d+n}\left(\sum_{i=-d}^{d} x^i r_{g^{x^i}} + \beta x^i r_{h^{x^i}} + \alpha\beta x^i r_{h^{\alpha x^i}} + \sum_{\substack{i=-d\\ i\neq 0}}^{d} \alpha x^i r_{g^{\alpha x^i}}\right) = 0\ .\end{aligned} \tag{27}$$

Therefore β is a root $f(\mathrm{B}) = 0$ in $\mathsf{ComHelper}_2(X, x, \alpha, [R], [W_a], a, z, n)$ invoked by h_2. Since $\mathsf{h}_2([\tau], x, \alpha, V)$ returned $\perp$ we have that $f(\mathrm{B})$ must be the zero polynomial, i.e.,

$$\alpha(x-z)\left(\sum_{i=-d}^{d} x^i w_{ah^{x^i}} + \alpha x^i w_{ah^{\alpha x^i}}\right) - x^{-d+n}\left(\sum_{i=-d}^{d} x^i r_{h^{x^i}} + \alpha x^i r_{h^{\alpha x^i}}\right) = 0\ .$$

Plugging this into (27) we get that

$$\begin{aligned}\alpha(x-z)\left(\sum_{i=-d}^{d} x^i w_{ag^{x^i}} + \sum_{\substack{i=-d\\ i\neq 0}}^{d} \alpha x^i w_{ag^{\alpha x^i}}\right) + \alpha a\\ - x^{-d+n}\left(\sum_{i=-d}^{d} x^i r_{g^{x^i}} + \sum_{\substack{i=-d\\ i\neq 0}}^{d} \alpha x^i r_{g^{\alpha x^i}}\right) = 0\ .\end{aligned}$$

Therefore α is a root $f(\mathrm{A}) = 0$ in $\mathsf{ComHelper}_3(X, x, [R], [W_a], a, z, n)$ invoked by h_3. Since $\mathsf{h}_3([\tau], x, \beta, V)$ returned $\perp$ it means that $f(\mathrm{A})$ is the zero polynomial. In particular its A term is 0 i.e.

$$(x-z)\left(\sum_{i=-d}^{d} x^i w_{ag^{x^i}}\right) + a - x^{-d+n}\left(\sum_{\substack{i=-d\\ i\neq 0}}^{d} x^i r_{g^{\alpha x^i}}\right) = 0\ .$$

Therefore x is a root $f(X) = 0$ in $\mathsf{ComHelper}_1(X, [R], [W_a], a, z, n)$ invoked by h_1. Now, since $\mathsf{h}_1([\tau], x, \beta, g^x)$ returned $\perp$ we have that $f_1(X)$ is the zero polynomial, i.e.,

$$(X-z)\left(\sum_{i=-d}^{d} X^i w_{ag^{x^i}}\right) + a - X^{-d+n}\left(\sum_{\substack{i=-d\\ i\neq 0}}^{d} X^i r_{g^{\alpha x^i}}\right)\ .$$

is the zero polynomial. The above polynomial is an zero for any value of X. So, plugging in $X = z$ we get

$$a - z^{-d+n}\left(\sum_{\substack{i=-d\\ i\neq 0}}^{d} z^i r_{g^{\alpha x^i}}\right) = 0\ .$$

So,

$$a = \left(\sum_{\substack{i=n-2d \\ i \neq n-d}}^{n} z^i r_{g^{\alpha x^{i-n+d}}} \right).$$

Similarly, since τ is an accepting transcript, the equalities

$$e(W_b, h^{\alpha x}) e(g^b W_b^{yz}, h^{\alpha}) = e(R, h^{x^{-d+n}}) \ , \ e(W_t, h^{\alpha x}) e(g^t W_t^z, h^{\alpha}) = e(T, h)$$

hold. Using arguments similar to the ones we used above, we can show that

$$b = \left(\sum_{\substack{i=n-2d \\ i \neq n-d}}^{n} (yz)^i r_{g^{\alpha x^{i-n+d}}} \right) \ , \ t = \left(\sum_{\substack{i=-d \\ i \neq 0}}^{d} z^i t_{g^{\alpha x^i}} \right).$$

Next we can show that for the opening of commitments T_j, C_j, A_j (similar to how we derived the value for a above) for $j = 1, \ldots, M$

$$t_j = \left(\sum_{\substack{i=-d \\ i \neq 0}}^{d} z_j^i t_{jg^{\alpha x^i}} \right),$$

$$k_j = \left(\sum_{\substack{i=-d \\ i \neq 0}}^{d} y_j^i c_{jg^{\alpha x^i}} \right),$$

$$c_j = \left(\sum_{\substack{i=-d \\ i \neq 0}}^{d} z_j^{-i} c_{jg^{\alpha x^i}} \right),$$

$$a_j = \left(\sum_{\substack{i=-d \\ i \neq 0}}^{d} (y_j z_j)^i a_{jg^{\alpha x^i}} \right).$$

Also using that for $j = 1, \ldots, M$,

$$\begin{aligned} &\mathsf{WfHelper}_1(g^x, [C], [C_{j,w,1}, C_{j,w,2}], 2n+1) = \perp , \\ &\mathsf{WfHelper}_2(V, x, \alpha, [C], [C_{j,w,1}, C_{j,w,2}], 2n+1) = \perp , \\ &\mathsf{WfHelper}_3(X, x, [C], [C_{j,w,1}, C_{j,w,2}], 2n+1) = \perp , \end{aligned}$$

we can show that $c_{jg^{\alpha x^i}} = 0$ for $i <= 0$ and for $i > 2n+1$ and $c_{jg^{x^i}} = 0$, $c_{jh^{\alpha x^i}} = 0$, $c_{jh^{x^i}} = 0$ for all i. Similarly, we can show that $u_{jg^{\alpha x^i}} = 0$ for $i <= 0$ and for $i > n$ and $u_{jg^{x^i}} = 0$, $u_{jh^{\alpha x^i}} = 0$, $u_{jh^{x^i}} = 0$ for all i and $v_{jg^{\alpha x^i}} = 0$ for $i <= 0$ and for $i > n$ and $v_{jg^{x^i}} = 0$, $v_{jh^{\alpha x^i}} = 0$, $v_{jh^{x^i}} = 0$ for all i.

Hence $c_j = \left(\sum_{i=1}^{2n+1} z_j^{-i} c_{jg^{\alpha x^i}}\right)$. We also have that $t_j = (ya_j + z_j^{n+2} + z_j^{n+1} y_j - z_j^{2n+2} y_j)(c_j + 1) z_j^{-1} - y_j k_j - 1$. Therefore, for $j = 1, \ldots, M$

$$\left(\sum_{\substack{i=-d \\ i\neq 0}}^{d} z_j^i t_{jg^{\alpha x^i}}\right) = \left(y_j \left(\sum_{\substack{i=-d \\ i\neq 0}}^{d} (y_j z_j)^i a_{jg^{\alpha x^i}}\right) + z_j^{n+2} + z_j^{n+1} y - z_j^{2n+2} y_j\right) \cdot \left(\sum_{i=1}^{2n+1} z_j^{-i} c_{jg^{\alpha x^i}} + 1\right) z_j^{-1} - \left(\sum_{i=1}^{2n+1} y_j^i c_{jg^{\alpha x^i}}\right) y_j - 1 \, .$$

Since $\tau|_{z_j} \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{SnACSPf}}$, we have that $z_j \notin \mathsf{BadCh}(\tau|_{z_j})$. Therefore, $\mathsf{SZ}(f(Z), z_j)$ is `false` where f is as defined in $\mathsf{CheckBad}(\tau', z_j)$. Since we have here that $f(z_j) = 0$, the polynomial $f(Z)$ is the zero polynomial. In particular, its constant term is zero, i.e.,

$$\sum_{i=1}^{2n+1} y_j^{i+1} \left(a_{jg^{\alpha x^i}} c_{jg^{\alpha x^{i-1}}} - c_{jg^{\alpha x^{i-1}}}\right) + (c_{jg^{\alpha x^{n+1}}} - 1) + y_j (c_{jg^{\alpha x^n}} - c_{jg^{\alpha x^{2n+1}}}) = 0 \, .$$

Since $\tau|_{y_j} \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{SnACSPf}}$, we have that $y_j \notin \mathsf{BadCh}(\tau|_{y_j})$. Therefore, $\mathsf{SZ}(f(Y), y_j)$ is `false` where f is as defined in $\mathsf{CheckBad}(\tau', y_j)$. Since we have here that $f(y_j) = 0$, the polynomial $f(Y)$ is the zero polynomial. In particular, its constant term is zero and we have that for $j = \{1, \ldots, M\}$

$$c_{jg^{\alpha x}} = a_{jg^{\alpha x}} \, , \; c_{jg^{\alpha x^{n+1}}} = 1 \, , \; c_{jg^{\alpha x^n}} = c_{jg^{\alpha x^{2n+1}}} \, ,$$

for $i = 2, \ldots, 2n+1$

$$c_{jg^{\alpha x^i}} = c_{jg^{\alpha x^{i-1}}} a_{jg^{\alpha x^i}} \, .$$

Combining, we get that

$$\prod_{i=1}^{n} a_{jg^{\alpha x^i}} = \prod_{i=n+2}^{2n+1} a_{jg^{\alpha x^i}} \, . \tag{28}$$

We also have that

$$e(g^{\alpha a_{j,n+1} x^{n+1}} U_j, h) e(V_j, h^{x^{n+1}}) = e(A_j, h)$$

Using that for $j = 1, \ldots, M$,

$$\begin{aligned} &\mathsf{EqHelper}_1(X, [U_i], [V_i], [A_i], a_{i,n+1}, n) = \perp \, , \\ &\mathsf{EqHelper}_2(X, x, \alpha, [U_i], [V_i], [A_i], a_{i,n+1}, n) = \perp \, , \\ &\mathsf{EqHelper}_3(X, x, [U_i], [V_i], [A_i], a_{i,n+1}) = \perp \, , \end{aligned}$$

we can show that

$$a_{jg^{\alpha x^i}} = \begin{cases} u_{jg^{\alpha x^i}} & \text{for } 1 \leqslant i \leqslant n \\ v_{jg^{\alpha x^{i-n-1}}} & \text{for } n+2 \leqslant i \leqslant 2n+1 \end{cases} \, .$$

Combining with (28) we have that

$$\prod_{i=1}^{n} u_{jg^{\alpha x^i}} = \prod_{i=1}^{n} v_{jg^{\alpha x^i}} \, .$$

Now from the definition of U_j, V_j we have that $u_{jg^{\alpha x^i}} = \alpha s_{jg^{\alpha x^i}} x^i + \beta_j \alpha \sigma_{j,i} x^i + \gamma_j \alpha x^i$ and $v_{jg^{\alpha x^i}} = \alpha s'_{g^{\alpha x^i}} x^i + \beta \alpha i x^i + \gamma_j \alpha x^i$. Therefore,

$$\prod_{i=1}^{n} \left(\alpha s_{jg^{\alpha x^i}} x^i + \beta_j \alpha \sigma_{j,i} x^i + \gamma_j \alpha x^i \right) = \prod_{i=1}^{n} \left(\alpha s'_{g^{\alpha x^i}} x^i + \beta \alpha i x^i + \gamma_j \alpha x^i \right)$$

Simplifying we get,

$$\prod_{i=1}^{n} \left(s_{jg^{\alpha x^i}} + \beta_j \sigma_{j,i} + \gamma_j \right) = \prod_{i=1}^{n} \left(s'_{jg^{\alpha x^i}} + \beta_j i + \gamma_j \right)$$

Since $\tau|_{(\beta_j,\gamma_j)} \notin \mathcal{T}^{\mathsf{SnACSPf}}_{\mathsf{BadCh}}$, we have that $(\beta_j, \gamma_j) \notin \mathsf{BadCh}(\tau|_{(\beta_j,\gamma_j)})$. Therefore, $\mathsf{SZ}(f(\mathrm{B}, \Gamma), (\beta_j, \gamma_j))$ is `false` where f is as defined in $\mathsf{CheckBad}(\tau', (\beta_j, \gamma_j))$. Since we have here that $f(\beta_j, \gamma_j) = 0$, the polynomial $f(\mathrm{B}, \Gamma)$ is the zero polynomial. Hence we have that for $j = \{1, \ldots, M\}$

$$s_{jg^{\alpha x^i}} = s_{jg^{\alpha x^{\sigma_i}}} \ .$$

Additionally we can infer the following from the opening of the commitments W'_j, Q'_j (similar to how we derived the value for a above)

$$v_j = \left(\sum_{\substack{i=-d \\ i \neq 0}}^{d} \delta_j^i s'_{g^{\alpha x^i}} \right),$$

$$v_j = \left(\sum_{\substack{i=-d \\ i \neq 0}}^{d} (\delta_j y_j)^i \psi_i \right).$$

Equating the two values of v_j we get that

$$\left(\sum_{\substack{i=-d \\ i \neq 0}}^{d} \delta_j^i s'_{g^{\alpha x^i}} \right) = \left(\sum_{\substack{i=-d \\ i \neq 0}}^{d} (\delta_j y_j)^i \psi_i \right).$$

Since $\tau|_{\delta_j} \notin \mathcal{T}^{\mathsf{SnACSPf}}_{\mathsf{BadCh}}$, we have that $\delta_j \notin \mathsf{BadCh}(\tau|_{\delta_j})$. Therefore, $\mathsf{SZ}(f(\Delta), \delta_j)$ is `false` where f is as defined in $\mathsf{CheckBad}(\tau', \delta_j)$. Since we have here that $f(\delta_j) = 0$, the polynomial $f(Y)$ is the zero polynomial. In particular, its constant term is zero and we have that for $j = \{1, \ldots, M\}$

$$s'_{jg^{\alpha x^i}} = \psi_i y^i \text{ for } i = 1, \ldots, n \ ,$$

and $s'_{jg^{\alpha x^i}} = 0$ for $i <= 0$ and $i > n$. Since we derived $s_{jg^{\alpha x^i}} = s_{jg^{\alpha x^{\sigma_i}}}$ above, we have that

$$s_{jg^{\alpha x^i}} = \psi_{\sigma_i} y^{\sigma_i} \ ,$$

and $s_{jg^{\alpha x^i}} = 0$ for $i <= 0$ and $i > n$. Additionally we can infer the following from the opening of the commitment W_j (similar to how we derived the value for a above)

$$s_j = \left(\sum_{\substack{i=-d \\ i \neq 0}}^{d} z^i s_{jg^{\alpha x^i}} \right).$$

Plugging in the values of $s_{jg^{\alpha x^i}}$

$$s_j = \left(\sum_{i=1}^{n} z^i \psi_{\sigma_i} y^{\sigma_i} \right) .$$

Since $s(X,Y) = \sum_{j=1}^{M} \sum_{i=1}^{n} X^i \psi_{\sigma_{j,i}} Y^{\sigma_{j,i}}$, we have that $s = \sum_{i=1}^{M} s_i = s(z,y)$. So, we have that

$$t = a(b + s(z,y)) - k(y) .$$

Plugging the values of $a, b, t, s(z,y)$ we get that

$$\left(\sum_{\substack{i=-d \\ i\neq 0}}^{d} z^i t_{g^{\alpha x^i}} \right) = \left(\sum_{\substack{i=n-2d \\ i\neq n-d}}^{n} z^i r_{g^{\alpha x^{i-n+d}}} \right) \left(\sum_{\substack{i=n-2d \\ i\neq n-d}}^{n} (yz)^i r_{g^{\alpha x^{i-n+d}}} + s(z,y) \right) - k(y)$$

Since $\tau|_z \notin \mathcal{T}^{\mathsf{SnACSPf}}_{\mathsf{BadCh}}$, we have that $z \notin \mathsf{BadCh}(\tau|_z)$. Therefore, $\mathsf{SZ}(f(Z), z)$ is `false` where f is as defined in $\mathsf{CheckBad}(\tau', z)$. Since we have here that $f(z) = 0$, the polynomial $f(Z)$ must be the zero polynomial. In particular, its constant term must be zero. Writing out the constant term of $f(Z)$ and using $\mathbf{a}^*_L = (r_{g^{\alpha x^{1-n+d}}}, \ldots, r_{g^{\alpha x^d}})$, $\mathbf{a}^*_R = (r_{g^{\alpha x^{-1-n+d}}}, \ldots, r_{g^{\alpha x^{d-2n}}})$ and $\mathbf{a}^*_O = (r_{g^{\alpha x^{-1-2n+d}}}, \ldots, r_{g^{\alpha x^{d-3n}}})$ we get

$$r_{g^{\alpha x^{d-n}}} r_{g^{\alpha x^{d-n}}} + \langle \mathbf{a}^*_L \circ \mathbf{a}^*_R - \mathbf{a}^*_O, \mathbf{y}^{n+1}_{[1:]} + \mathbf{y}^{-n-1}_{[1:]} \rangle + \mathbf{y}^{Q+n+1}_{[n+1:]} \cdot (\mathbf{W}_L \cdot \mathbf{a}^*_L + \mathbf{W}_R \cdot \mathbf{a}^*_R + \mathbf{W}_O \cdot \mathbf{a}^*_O) - \langle \mathbf{c}, \mathbf{y}^{Q+n+1}_{[n+1:]} \rangle = 0$$

Since $\tau|_y \notin \mathcal{T}^{\mathsf{SnACSPf}}_{\mathsf{BadCh}}$ we have that $y \notin \mathsf{BadCh}(\tau|_y)$. Therefore, $\mathsf{SZ}(f(Y), y)$ is `false` where f is as defined in $\mathsf{CheckBad}(\tau', y)$. Since we have here that $f(y) = 0$, the polynomial $f(Y)$ is the zero polynomial. Therefore, equating all the coefficients of $f(Y)$ to zero, we have that

$$\mathbf{a}^*_L \circ \mathbf{a}^*_R = \mathbf{a}^*_O \text{ and } \mathbf{a}^*_L \cdot \mathbf{W}_L + \mathbf{a}^*_R \cdot \mathbf{W}_R + \mathbf{a}^*_O \cdot \mathbf{W}_O = \mathbf{c} .$$

□

7 Non-adaptive srs-wee security

The notion of srs-wee security that we defined in Section 4 allows the prover to adaptively choose its input. In the srs-wee analysis the extractor had access to the representation of the instance since the algebraic prover generated the instance. But there are scenarios where the prover may be non-adaptive and not be able to do that – for example, the input could be generated by another party, and the prover tries to prove knowledge with respect to this input. Hence this section, we consider a setting where instead of the prover adaptively choosing its input, an instance generator generates an instance which is given to the prover. For protocols where the instance contains group elements, we need an analysis entirely different from the analysis for srs-ewe because here the representation of the instance will not available to the extractor.

Game $\mathsf{NWEE\text{-}1}^{\mathcal{P}_{\mathsf{alg}},\mathcal{D}}_{\mathsf{IP},\mathsf{Gen}}(\lambda)$:

$\mathsf{tr} \leftarrow \varepsilon$
$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{IP.Setup}(1^\lambda)$
$x \leftarrow\!\!\$\ \mathsf{Gen}(\mathsf{pp})$
Run $\mathcal{P}^{\mathbf{O}^1_{\mathrm{ext}}}_{\mathsf{alg},\lambda}(\mathsf{pp}, x)$
$b \leftarrow\!\!\$\ \mathcal{D}(\mathsf{tr})$
Return $(b = 1)$

Game $\mathsf{NWEE\text{-}0}^{\mathcal{E},\mathcal{P}_{\mathsf{alg}},\mathcal{D}}_{\mathsf{IP},R,\mathsf{Gen}}(\lambda)$:

$\mathsf{tr} \leftarrow \varepsilon$
$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{IP.Setup}(1^\lambda)$
$x \leftarrow\!\!\$\ \mathsf{Gen}(\mathsf{pp})$
$\mathsf{st}_{\mathcal{E}} \leftarrow (1^\lambda, \mathsf{pp}, x)$
Run $\mathcal{P}^{\mathbf{O}^0_{\mathrm{ext}}}_{\mathsf{alg},\lambda}(\mathsf{pp}, x)$
$w' \leftarrow\!\!\$\ \mathcal{E}(\mathsf{st}_{\mathcal{E}}, \bot)$
$b \leftarrow\!\!\$\ \mathcal{D}(\mathsf{tr})$
Return $(b = 1) \wedge (\mathsf{Acc}(\mathsf{tr}) \Rightarrow (\mathsf{pp}, x, w') \in R)$

Oracle $\mathbf{O}^1_{\mathrm{ext}}(\tau = (a_1, c_1, \ldots, a_{i-1}, c_{i-1}), a_i)$:

If $\tau \in \mathsf{tr}$ then
  If $i \leqslant r$ then
    $c_i \leftarrow\!\!\$\ \mathsf{Ch}_i$; $\mathsf{tr} \leftarrow \mathsf{tr} \,\|\, (\tau, a_i, c_i)$; return c_i
  Else if $i = r + 1$ then
    $d \leftarrow \mathsf{IP.V}(\mathsf{pp}, x, \tau \,\|\, a_i)$
    If $d = 1$ then return d
Return $\bot$

Oracle $\mathbf{O}^0_{\mathrm{ext}}(\tau = (a_1, c_1, \ldots, a_{i-1}, c_{i-1}), a_i)$:

If $\tau \in \mathsf{tr}$ then
  If $i \leqslant r$ then
    $(\mathsf{resp}, \mathsf{st}_{\mathcal{E}}) \leftarrow\!\!\$\ \mathcal{E}(\mathsf{st}_{\mathcal{E}}, [(\tau, a_i)])$
  $\mathsf{tr} \leftarrow \mathsf{tr} \,\|\, (\tau, a_i, \mathsf{resp})$
  Return resp
  Return $\bot$

Fig. 30. Definition of the security notion n-srs-wee. The games NWEE-1, NWEE-0 define n-srs-wee security in the AGM for a non-uniform algebraic prover $\mathcal{P}_{\mathsf{alg}}$, a distinguisher $\mathcal{D}$, an extractor $\mathcal{E}$, generator Gen and a public-coin interactive proof IP. We assume here that IP has $r = r(\lambda)$ challenges and the i-th challenge is sampled from Ch_i.

FORMALIZING N-SRS-WEE SECURITY. We formalize another notion of proof-of-knowledge (PoK) security in the AGM where the input is not generated by the prover, instead the prover is given an instance generated by an instance generator (we assume that Gen is an algorithm that takes as input the public parameters and returns an instance x). We give a definition along the lines of srs-wee. This new security notion called non-adaptive srs-wee or n-srs-wee is formally defined using games NWEE-0, NWEE-1 in Figure 30. For an interactive proof IP, an associated relation R and an instance generator Gen, non-uniform algebraic prover $\mathcal{P}_{\mathsf{alg}}$, a distinguisher $\mathcal{D}$, and an extractor $\mathcal{E}$, we define

$$\mathsf{Adv}^{\mathsf{n\text{-}sr\text{-}wee}}_{\mathsf{IP},R,\mathsf{Gen}}(\mathcal{P}_{\mathsf{alg}}, \mathcal{D}, \mathcal{E}, \lambda) = \Pr\left[\mathsf{NWEE\text{-}1}^{\mathcal{P}_{\mathsf{alg}},\mathcal{D}}_{\mathsf{IP},\mathsf{Gen}}(\lambda)\right] - \Pr\left[\mathsf{NWEE\text{-}0}^{\mathcal{E},\mathcal{P}_{\mathsf{alg}},\mathcal{D}}_{\mathsf{IP},R,\mathsf{Gen}}(\lambda)\right] . \tag{29}$$

N-FS-EXT SECURITY. We can formalize a notion of non-adaptive proof-of-knowledge (PoK) security in the AGM for non-interactive arguments obtained by applying the Fiat-Shamir transform to an interactive protocol IP analogous to fs-ext-1 security. We can define it through a game N-FS-EXT whose only difference from the game FS-EXT-1 is that there is an instance generator Gen which outputs an instance x and the prover has to output a proof for the instance x instance of being able to choose it adaptively. For an interactive proof IP and an associated relation R, instance generator Gen algebraic prover $\mathcal{P}_{\mathsf{alg}}$ and an extractor $\mathcal{E}$, we define

$$\mathsf{Adv}^{\mathsf{n\text{-}fs\text{-}ext}}_{\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}],\mathsf{Gen},R}(\mathcal{P}_{\mathsf{alg}}, \mathcal{E}, \lambda) = \Pr\left[\mathsf{N\text{-}FS\text{-}EXT}^{\mathcal{P}_{\mathsf{alg}},\mathcal{E}}_{\mathsf{IP},R}(\lambda)\right] .$$

The following theorem connects the n-srs-wee of a public-coin protocol IP and the n-fs-ext soundness of non-interactive protocol $\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}]$, obtained by applying the Fiat-Shamir transform using a random oracle. Its proof is very similar to the proof of Theorem 2 and has been omitted.

Theorem 8. *Let R be a relation. Let* IP *be a $r = r(\lambda)$-challenge public coin interactive protocol for the relation R where the length of the i^{th} challenge is* $\mathsf{cLen}_i(\lambda)$ *such that* $\mathsf{sLen}(\lambda) \leqslant \mathsf{cLen}_i(\lambda) \leqslant \mathsf{hLen}(\lambda)$

for $i \in \{1, \ldots, r\}$. Let Gen *be an instance generator for the relation R. Let $\mathcal{E}$ be an extractor for* IP *such that it always responds to queries with bit-strings of appropriate length chosen uniformly at random. We can construct an extractor $\mathcal{E}^*$ for* $\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}]$ *such that for every non-uniform algebraic prover $\mathcal{P}^*_{\mathsf{alg}}$ against* $\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}]$ *that makes $q = q(\lambda)$ random oracle queries, there exists a non-uniform algebraic prover $\mathcal{P}_{\mathsf{alg}}$ and $\mathcal{D}$ such that for all $\lambda \in \mathbb{N}^+$*

$$\mathsf{Adv}^{\mathsf{n\text{-}fs\text{-}ext}}_{\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}],R}(\mathcal{P}^*_{\mathsf{alg}}, \mathcal{E}, \lambda) \leqslant \mathsf{Adv}^{\mathsf{n\text{-}sr\text{-}wee}}_{\mathsf{IP},R}(\mathcal{P}_{\mathsf{alg}}, \mathcal{D}, \mathcal{E}, \lambda) + \frac{q+1}{2^{\mathsf{sLen}(\lambda)}} \ .$$

Moreover, $\mathcal{P}_{\mathsf{alg}}$ makes at most q queries to its oracle and is nearly as efficient as $\mathcal{P}^*_{\mathsf{alg}}$. The extractor $\mathcal{E}^*$ is nearly as efficient as $\mathcal{E}$.

N-SRS-WEE SECURITY OF RngPf. Among the protocols whose srs-wee security we analysed earlier, the only protocol which had a group element in its input was RngPf. The protocols $\mathsf{ACSPf}, \mathsf{SnACSPf}$ do not have a group element in their input and therefore their n-srs-wee security analysis would be identical to the analysis for srs-wee security. For RngPf however, since we needed to use the representation of the element V, which is not available in the n-srs-wee setting, we need to give a new analysis. Next, in the following theorem we analyse the n-srs-wee security of RngPf.

Theorem 9. *Let $\mathbb{G} = \{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}^+}$ be a family of groups of order $p = p(\lambda)$. Let* $\mathsf{RngPf} = \mathsf{RngPf}[\mathbb{G}]$ *be the interactive argument as defined in Figure 7, for the relation R in (4). Let* Gen *be an instance generator for the relation R. We can construct an extractor $\mathcal{E}$ such that for any non-uniform algebraic prover $\mathcal{P}_{\mathsf{alg}}$ making at most $q = q(\lambda)$ queries to its oracle, there exists a non-uniform adversary $\mathcal{F}$ with the property that for any (computationally unbounded) distinguisher $\mathcal{D}$, for all $\lambda \in \mathbb{N}^+$*

$$\mathsf{Adv}^{\mathsf{n\text{-}sr\text{-}wee}}_{\mathsf{RngPf},R,\mathsf{Gen}}(\mathcal{P}_{\mathsf{alg}}, \mathcal{D}, \mathcal{E}, \lambda) \leqslant \sqrt{\mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{F}) + \frac{2q(14n+8)}{p-1} + \frac{1}{p}} \ .$$

Moreover, the time complexity of the extractor $\mathcal{E}$ is $O(q \cdot n)$ and that of adversary $\mathcal{F}$ is $O(q \cdot n)$.

Using Theorem 8, we get the following corollary.

Corollary 4. *Let $\mathbb{G} = \{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}^+}$ be a family of groups of order $p = p(\lambda)$. Let* $\mathsf{RngPf} = \mathsf{RngPf}[\mathbb{G}]$ *be the interactive argument as defined in Figure 7, for the relation R in (4). Let* Gen *be an instance generator for the relation R. Let* $\mathsf{FS}^{\mathbf{RO}}[\mathsf{RngPf}]$ *be the non-interactive argument obtained by applying the Fiat-Shamir transform to* RngPf *using a random oracle. We can construct an extractor $\mathcal{E}$ such that for any non-uniform algebraic prover $\mathcal{P}_{\mathsf{alg}}$ making at most $q = q(\lambda)$ queries to the random oracle there exists a non-uniform adversary $\mathcal{F}$ with the property that for all $\lambda \in \mathbb{N}^+$*

$$\mathsf{Adv}^{\mathsf{n\text{-}fs\text{-}ext}}_{\mathsf{FS}^{\mathbf{RO}}[\mathsf{RngPf}],\mathsf{Gen},R}(\mathcal{P}_{\mathsf{alg}}, \mathcal{E}, \lambda) \leqslant \sqrt{\mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{F}) + \frac{2q(14n+8)}{p-1} + \frac{1}{p}} + \frac{q+1}{p-1} \ .$$

Moreover, the time complexity of the extractor $\mathcal{E}$ is $O(q \cdot n)$ and that of adversary $\mathcal{F}$ is $O(q \cdot n)$.

We cannot reuse the framework we developed in section 4 to prove theorem 9 because the representation of V is not available. However, at the algebraic level this proof shares similarities with the proof of Theorem 4. So we provide a proof sketch omitting some details which are similar to that in Theorem 4.

Procedure $\mathsf{e}([\tau])$:

$//[\tau] = \big((n, \mathbf{g}, \mathbf{h}, u, g, h), [V]\,; ([A]\,, [S]), (y, z), ([T_1]\,, [T_2]), x, (\beta_x, \mu, \hat{t}), w, ([L_1]\,, [R_1]), x_1, \ldots,$
$\quad ([L_{\log n}]\,, [R_{\log n}]), x_{\log n}, (a, b)\big)$

If $z^2 + t_{1V}x + t_{2V}x^2 = 0$ then return $\perp$

$\delta(y, z) \leftarrow (z - z^2)\langle \mathbf{1}^n, \mathbf{y}^n \rangle - z^3 \langle \mathbf{1}^n, \mathbf{2}^n \rangle$

$v^* \leftarrow (\hat{t} - \delta(y, z) - t_{1g}x - t_{2g}x^2)(z^2 + t_{1V}x + t_{2V}x^2)^{-1}$

$\gamma^* \leftarrow (\beta_x - t_{1h}x - t_{2h}x^2)(z^2 + t_{1V}x + t_{2V}x^2)^{-1}$

Return (v^*, γ^*)

Fig. 31. The function e for RangeProof.

Proof (Sketch).

Like previously, we construct an extractor $\mathcal{E}$ that just answers challenges honestly, and applies the function e (defined in Figure 31) to a path in the execution tree which defines an accepting transcript, and returns whatever e returns. Observe from the definition of RngPf.V that if τ as defined in (5) is an accepting transcript,

$$V^{z^2} g^{\delta(y,z)} T_1^x T_2^{x^2} = g^{\hat{t}} h^{\beta_x} \ .$$

Now, e can plug in the representations of T_1, T_2 into the above equation and compute the values $e_V, e_{\mathbf{g}}, e_{\mathbf{h}}, e_u, e_g, e_h$ such that $V^{e_V} = \mathbf{g}^{e_{\mathbf{g}}} \mathbf{h}^{e_{\mathbf{h}}} u^{e_u} g^{e_g} h^{e_h}$. For example

$$e_g = \hat{t} - \delta(y, z) - t_{1g}x - t_{2g}x^2 \ , \ e_V = z^2 + t_{1V}x + t_{2V}x^2 \ .$$

The procedure e returns e_g/e_V and e_h/e_V. However, its output is a valid witness only if $e_{\mathbf{g}} = e_{\mathbf{h}} = \mathbf{0}^n, e_u = 0$ and $e_g/e_V \in [0, 2^n - 1]$.

In order to upper bound the failure probability of e, we shall again construct an adversary $\mathcal{H}$ against the discrete logarithm relation problem. In this case, it would first run Gen with its inputs and get V. Like previously, it would run $\mathcal{P}_{\mathsf{alg}}$ on $n, \mathbf{g}, \mathbf{h}, g, h, u, V$. If $\mathcal{P}_{\mathsf{alg}}$ manages to produce an accepting transcript τ_1, $\mathcal{H}$ runs $\mathcal{P}_{\mathsf{alg}}$ on $n, \mathbf{g}, \mathbf{h}, g, h, u, V$ with fresh randomness. The adversary $\mathcal{H}$ needs to re-run $\mathcal{P}_{\mathsf{alg}}$ twice because the representation of V is not available.

If $\mathcal{P}_{\mathsf{alg}}$ manages to produce an accepting transcript τ_2, $\mathcal{H}$ calls a helper function h on input $([\tau_1]\,, [\tau_2])$ and outputs whatever h outputs. The definition of h is given in Figure 32.

Define E as the event that $\mathcal{P}_{\mathsf{alg}}$ succeeds in producing an accepting transcript but $\mathcal{E}$ fails. Let $\mathsf{Pr}[E] = \delta$. Let $\delta_V = \mathsf{Pr}[E | V \text{ was output by Gen}]$. It follows that $\delta = \mathsf{E}[\delta_V]$ where the expectation is over all V output by Gen.

We shall show that

$$\mathsf{Adv}^{\mathsf{dl\text{-}rel}}_{\mathbb{G},2n+3}(\mathcal{H}) \geqslant \mathsf{E}\left[\delta_V^2\right] - \frac{2q(14n+8)}{p-1} \ .$$

Using Lemma 2 we would have that there exists an adversary $\mathcal{F}$ such that

$$\mathsf{E}\left[\delta_V^2\right] \leqslant \mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{F}) + \frac{2q(14n+8)}{p-1} + \frac{1}{p} \ .$$

Using Jensen's inequality we have that $\mathsf{E}\left[\delta_V^2\right] \geqslant (\mathsf{E}[\delta_V])^2 = \delta^2$.

So, we have that

$$\delta \leqslant \sqrt{\mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{F}) + \frac{2q(14n+8)}{p-1} + \frac{1}{p}} \ .$$

Then, it is easy to see that

$$\mathsf{Adv}^{\mathsf{n\text{-}sr\text{-}wee}}_{\mathsf{RngPf},R,\mathsf{Gen}}(\mathcal{P}_{\mathsf{alg}},\mathcal{D},\mathcal{E},\lambda) \leqslant \sqrt{\mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{F}) + \frac{2q(14n+8)}{p-1} + \frac{1}{p}} \,.$$

To conclude the proof of the theorem we need to show that

$$\mathsf{Adv}^{\mathsf{dl\text{-}rel}}_{\mathbb{G},2n+3}(\mathcal{H}) \geqslant \mathsf{E}\left[\delta_V^2\right] - \frac{2q(14n+8)}{p-1} \,.$$

Suppose $\mathcal{H}$ runs $\mathcal{P}_{\mathsf{alg}}$, and it succeeds in producing accepting transcripts τ_1, τ_2 where for $i = 1, 2$

$$[\tau_i] = \big((n, \mathbf{g}, \mathbf{h}, u, g, h), V; ([A_i], [S_i]), (y_i, z_i), ([T_{i1}], [T_{i2}]), x_i, (\beta_{ix}, \mu_i, \hat{t}_i), w_i, \\ ([L_{i1}], [R_{i1}]), x_{i1}, \ldots, ([L_{i\log n}], [R_{i\log n}]), x_{i\log n}, (a_i, b_i)\big) \,.$$

Define for $i = 1, 2$,

$$\begin{aligned}
\delta(y_i, z_i) &= (z_i - z_i^2)\langle \mathbf{1}^n, {\mathbf{y_i}}^n \rangle - z_i^3 \langle \mathbf{1}^n, \mathbf{2}^n \rangle \,, \\
v_i^* &= (\hat{t}_i - \delta(y_i, z_i) - t_{i1g} x_i - t_{i2g} x_i^2)(z_i^2 + t_{i1V} x_i + t_{i2V} x_i^2)^{-1} \,, \\
\gamma_i^* &= (\beta_{ix} - t_{i1h} x_i - t_{i2h} x_i^2)(z_i^2 + t_{i1V} x_i + t_{i2V} x_i^2)^{-1} \,.
\end{aligned}$$

Let E_1 be the event that $\mathcal{H}$ runs $\mathcal{P}_{\mathsf{alg}}$, and it succeeds in producing accepting transcripts τ_1, τ_2 and both of the following are true

- $v_1^* \notin [0, 2^n - 1] \vee g^{v_1^*} h^{\gamma_1^*} \neq V$
- $v_2^* \notin [0, 2^n - 1] \vee g^{v_2^*} h^{\gamma_2^*} \neq V$

It is easy to see that $\mathsf{Pr}\left[E_1\right] = \mathsf{E}\left[\delta_V^2\right]$. We define the pair of transcripts (τ_1, τ_2) to be bad if any of the following is true.

- $\mathsf{CheckBad}([\tau_i|_{w_i}], w_i)$ is `false` for $i = 1, 2$ where $\mathsf{CheckBad}([\tau'], w)$ is defined in Figure 8.
- $\mathsf{CheckBad}([\tau_i|_{x_{im}}], x_{im})$ is `false` for $i = 1, 2$ and $m = 1, \ldots, \log n$ where $\mathsf{CheckBad}([\tau'], x_m)$ is defined in Figure 8.
- $\mathsf{CheckBad}([\tau_1], [\tau_2], x_1, x_2)$ is `false` where $\mathsf{CheckBad}([\tau_1], [\tau_2], x_1, x_2)$ is defined in Figure 33.
- $\mathsf{CheckBad}([\tau_1], [\tau_2], (y_1, y_2), (z_1, z_2))$ is `false` where $\mathsf{CheckBad}([\tau_1], [\tau_2], (y_1, y_2), (z_1, z_2))$ is defined in Figure 33.

It can be shown that if $\mathcal{P}_{\mathsf{alg}}$ makes at most q queries in each execution, the probability that the transcripts (τ_1, τ_2) are bad is at most $\frac{2q(14n+8)}{p-1}$. The proof of this statement is similar to the proof of 5 and we omit the proof here.

First it is easy to see that the output h is always a discrete logarithm relation between $\mathbf{g}, \mathbf{h}, u, g, h$ but might be a trivial relation. Now we shall show that for a fixed (τ_1, τ_2), $\mathsf{h}(\tau_1, \tau_2)$ returns a trivial discrete logarithm relation and transcripts (τ_1, τ_2) are not bad then the event E_1 cannot happen. In other words, if E_1 happens then either h returns a non-trivial discrete logarithm relation or transcripts (τ_1, τ_2) are bad i.e.,

$$\mathsf{Pr}\left[E_1\right] \leqslant \mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{F}) + \frac{2q(14n+8)}{p-1} \,.$$

Procedure $\mathsf{h}([\tau_1],[\tau_2])$:

$//[\tau_i] = \big((n,\mathbf{g},\mathbf{h},u,g,h),V;([A_i],[S_i]),(y_i,z_i),([T_{i1}],[T_{i2}]),x_i,(\beta_{ix},\mu_i,\hat{t}_i),w_i,([L_{i1}],[R_{i1}]),x_{i1},\ldots,$
$([L_{i\log n}],[R_{i\log n}]),x_{i\log n},(a_i,b_i)\big)$

$\delta(Y,Z) \leftarrow (Z-Z^2)\langle \mathbf{1}^n, \mathbf{Y}^n\rangle - Z^3\langle \mathbf{1}^n, \mathbf{2}^n\rangle$

$e_{\mathbf{g}}^{(1)} \leftarrow (z_2^2 + t_{21V}x_2 + t_{22V}x_2^2)(t_{11\mathbf{g}}x_1 + t_{12\mathbf{g}}x_1^2) - (z_1^2 + t_{11V}x_1 + t_{12V}x_1^2)(t_{21\mathbf{g}}x_2 + t_{22\mathbf{g}}x_2^2)$; $e_{\mathbf{h}}^{(1)} \leftarrow (z_2^2 + t_{21V}x_2 + t_{22V}x_2^2)(t_{11\mathbf{h}}x_1 + t_{12\mathbf{h}}x_1^2) - (z_1^2 + t_{11V}x_1 + t_{12V}x_1^2)(t_{21\mathbf{h}}x_2 + t_{22\mathbf{h}}x_2^2)$

$e_u^{(1)} \leftarrow (z_2^2 + t_{21V}x_2 + t_{22V}x_2^2)(t_{11u}x_1 + t_{12u}x_1^2) - (z_1^2 + t_{11V}x_1 + t_{12V}x_1^2)(t_{21u}x_2 + t_{22u}x_2^2)$; $e_g^{(1)} \leftarrow (z_2^2 + t_{21V}x_2 + t_{22V}x_2^2)(\delta(y_1,z_1) + t_{11g}x_1 + t_{12g}x_1^2 - \hat{t}_1) - (z_1^2 + t_{11V}x_1 + t_{12V}x_1^2)(\delta(y_2,z_2) + t_{21g}x_2 + t_{22g}x_2^2 - \hat{t}_2)$

$e_h^{(1)} \leftarrow (z_2^2 + t_{21V}x_2 + t_{22V}x_2^2)(t_{11h}x_1 + t_{12h}x_1^2 - \beta_{1x}) - (z_1^2 + t_{11V}x_1 + t_{12V}x_1^2)(t_{21h}x_2 + t_{22h}x_2^2 - \beta_{2x})$

If $(e_{\mathbf{g}}^{(1)}, e_{\mathbf{h}}^{(1)}, e_u^{(1)}, e_g^{(1)}, e_h^{(1)}) \neq (\mathbf{0}^n, \mathbf{0}^n, 0, 0, 0)$ then return $(e_{\mathbf{g}}^{(1)}, e_{\mathbf{h}}^{(1)}, e_u^{(1)}, e_g^{(1)}, e_h^{(1)})$

For $i = 1, 2$ do

$v_{ig} \leftarrow (\delta(y_i,z_i) + t_{i1g}x_i + t_{i2g}x_i^2 - \hat{t}_i)(z_i^2 + t_{i1V}x_i + t_{i2V}x_i^2)^{-1}$

$v_{ih} \leftarrow (t_{i1h}x_i + t_{i2h}x_i^2 - \beta_{ix})(z_i^2 + t_{i1V}x_i + t_{i2V}x_i^2)^{-1}$

$p'_{iV} \leftarrow a_{iV} + x_i s_{iV}$; $p'_{i\mathbf{g}} \leftarrow (a_{i\mathbf{g}}) + x_i s_{i\mathbf{g}} - z_i \mathbf{1}^n$

$p'_{i\mathbf{h}} \leftarrow a_{i\mathbf{h}} + x s_{i\mathbf{h}} + \mathbf{y_i}^{-n} \circ (z_i \mathbf{y_i}^n + z_i^2 \mathbf{2}^n)$; $p'_{ig} \leftarrow a_{ig} + x_i s_{ig}$

$p'_{ih} \leftarrow a_{ih} + x_i s_{ih} - \mu_i$; $p'_{iu} \leftarrow a_{iu} + x_i s_{iu} + w_i \hat{t}_i$

For $k = 0$ to $n-1$ do

$e_{g_{k+1}}^{(i,2)} \leftarrow \left(p'_{ig_{1+k}} + \sum_{m=1}^{\log n} l_{img_{1+k}} x_m^2 + r_{img_{1+k}} x_m^{-2}\right) - a \cdot \left(\prod_{m=1}^{\log n} x_m^{(-1)^{1-\mathsf{bit}(k,m,\log n)}}\right)$

$e_{h_{k+1}}^{(i,2)} \leftarrow \left(p'_{ih_{1+k}} + \sum_{m=1}^{\log n} l_{imh_{1+k}} x_m^2 + r_{imh_{1+k}} x_m^{-2}\right) - by^{(-(k))} \cdot \left(\prod_{m=1}^{\log n} x_m^{(-1)^{\mathsf{bit}(k,m,\log n)}}\right)$

$e_{\mathbf{g}}^{(i,2)} \leftarrow (e_{g_1}^{(i,2)}, \ldots, e_{g_n}^{(i,2)})$; $e_{\mathbf{h}}^{(i,2)} \leftarrow (e_{h_1}^{(i,2)}, \ldots, e_{h_n}^{(i,2)})$

$e_u^{(i,2)} \leftarrow \left(p'_{iu} + \sum_{m=1}^{\log n} l_{imu} x_m^2 + r_{imu} x_m^{-2}\right) - w_i \cdot a_i b_i$

$e_g^{(2)} \leftarrow \left(\sum_{m=1}^{\log n} (l_{img} + l_{imV} v_{ig}) x_m^2 + (r_{img} + r_{imV} v_{ig}) x_m^{-2}\right) + p'_{ig} + p'_{iV} v_{ig}$

$e_h^{(2)} \leftarrow \left(\sum_{m=1}^{\log n} (l_{imh} + l_{imV} v_{ih}) x_m^2 + (r_{imh} + r_{imV} v_{ih}) x_m^{-2}\right) + p'_{ih} + p'_{iV} v_{ih}$

If $(e_{\mathbf{g}}^{(1,2)}, e_{\mathbf{h}}^{(1,2)}, e_u^{(1,2)}, e_g^{(1,2)}, e_h^{(1,2)}) \neq (\mathbf{0}^n, \mathbf{0}^n, 0, 0, 0)$ then return $(e_{\mathbf{g}}^{(1,2)}, e_{\mathbf{h}}^{(1,2)}, e_u^{(1,2)}, e_g^{(1,2)}, e_h^{(1,2)})$

Return $(e_{\mathbf{g}}^{(2,2)}, e_{\mathbf{h}}^{(2,2)}, e_u^{(2,2)}, e_g^{(2,2)}, e_h^{(2,2)})$

Fig. 32. The function h for RngPf.

This would give us

$$\mathsf{Adv}^{\mathsf{dl\text{-}rel}}_{\mathbb{G},2n+3}(\mathcal{H}) \geqslant \mathsf{E}\left[\delta_V^2\right] - \frac{2q(14n+8)}{p-1}\ .$$

We shall use the approach we used in the proof of Lemma 6 to prove that for a fixed (τ_1,τ_2), $\mathsf{h}(\tau_1,\tau_2)$ returns a trivial discrete logarithm relation and transcripts (τ_1,τ_2) are not bad then the event E_1 cannot happen. We shall first use that $(e_{\mathbf{g}}^{(1)}, e_{\mathbf{h}}^{(1)}, e_u^{(1)}, e_g^{(1)}, e_h^{(1)}) = (\mathbf{0}^n, \mathbf{0}^n, 0, 0, 0)$ and (τ_1,τ_2) is not bad to conclude that $g^{v_1^*} h^{\gamma_1^*} = V$ holds. First using $e_{\mathbf{g}}^{(1)} = \mathbf{0}^n$ we get that

$$(z_2^2 + t_{21V}x_2 + t_{22V}x_2^2)(t_{11\mathbf{g}}x_1 + t_{12\mathbf{g}}x_1^2) - (z_1^2 + t_{11V}x_1 + t_{12V}x_1^2)(t_{21\mathbf{g}}x_2 + t_{22\mathbf{g}}x_2^2) = 0$$

Since (τ_1,τ_2) is not a bad transcript we can show that $t_{11\mathbf{g}} = \mathbf{0}^n$, $t_{12\mathbf{h}} = \mathbf{0}^n$, $t_{21\mathbf{g}} = \mathbf{0}^n$, $t_{21\mathbf{h}} = \mathbf{0}^n$. Similarly using $e_{\mathbf{h}}^{(1)} = \mathbf{0}^n$ and $e_u^{(1)} = 0$ we can show that $t_{i1\mathbf{h}} = \mathbf{0}^n$, $t_{i2\mathbf{h}} = \mathbf{0}^n$, $t_{iu} = 0$ for $i = 1, 2$.

This means that we have that

$$V^{z_1^2 + t_{11v}x_1 + t_{12v}x_1^2} = g^{\hat{t}_1 - \delta(y_1,z_1) - t_{11g}x_1 - t_{12g}x_1^2} h^{\beta_{1x} - t_{11h}x_1 - t_{12h}x_1^2}\ .$$

Since (τ_1,τ_2) are not bad, we have that $z_1^2 + t_{11v}x_1 + t_{12v}x_1^2 \neq 0$, so $g^{v_1^*} h^{\gamma_1^*} = V$.

Procedure $\mathsf{CheckBad}([\tau_1], [\tau_2], (y_1, z_1), (y_2, z_2))$:

$f_1(Y,Z) \leftarrow Z^2(\delta(y_1,z_1)+t_{11g}x_1+t_{12g}x_1^2-\hat{t}_1)-(z_1^2+t_{11V}x_1+t_{12V}x_1^2)(\delta(Y,Z)-\langle a_{2\mathbf{g}} - Z\mathbf{1}^n, (a_{2\mathbf{h}} + Z\mathbf{1}^n) \circ \mathbf{Y}^n + Z^2\mathbf{2}^n\rangle)$
$f_2(Y,Z) \leftarrow Z^2(\langle a_{2\mathbf{g}}, \mathbf{2}^n\rangle - \langle a_{\mathbf{g}}, \mathbf{2}^n\rangle) - Z\langle a_{\mathbf{g}} - a_{\mathbf{h}} - \mathbf{1}^n, \mathbf{Y}^n\rangle - \langle a_{\mathbf{g}} \circ a_{\mathbf{h}}, \mathbf{Y}^n\rangle$
Return $\mathsf{SZ}(f_1(Z), z) \vee \mathsf{SZ}(f_2(Z), z)$

Procedure $\mathsf{CheckBad}([\tau_1'], [\tau_2'], x_1, x_2)$:

$//[\tau_i'] = ((n, \mathbf{g}, \mathbf{h}, u, g, h), V, ([A_i], [S_i]), (y_i, z_i), ([T_{i1}], [T_{i2}]))$
$\delta(Y,Z) \leftarrow (Z - Z^2)\langle \mathbf{1}^n, \mathbf{Y}^n\rangle - Z^3\langle \mathbf{1}^n, \mathbf{2}^n\rangle$
If $z_1^2 + t_{11V}x_1 + t_{12V}x_1^2 = 0$ then return `true`
If $z_2^2 + t_{21V}x_2 + t_{22V}x_2^2 = 0$ then return `true`
$f_{11}(X) \leftarrow (z_2^2 + t_{21V}X + t_{22V}X^2)(t_{11\mathbf{g}}x_1 + t_{12\mathbf{g}}x_1^2) - (z_1^2 + t_{11V}x_1 + t_{12V}x_1^2)(t_{21\mathbf{g}}X + t_{22\mathbf{g}}X^2)$
$f_{12}(X) \leftarrow (z_2^2 + t_{21V}x_2 + t_{22V}x_2^2)(t_{11\mathbf{g}}X + t_{12\mathbf{g}}X^2) - (z_1^2 + t_{11V}X + t_{12V}X^2)(t_{21\mathbf{g}}x_2 + t_{22\mathbf{g}}x_2^2)$
$f_{21}(X) \leftarrow (z_2^2 + t_{21V}X + t_{22V}X^2)(t_{11\mathbf{h}}x_1 + t_{12\mathbf{h}}x_1^2) - (z_1^2 + t_{11V}x_1 + t_{12V}x_1^2)(t_{21\mathbf{h}}X + t_{22\mathbf{h}}X^2)$
$f_{22}(X) \leftarrow (z_2^2 + t_{21V}x_2 + t_{22V}x_2^2)(t_{11\mathbf{h}}X + t_{12\mathbf{h}}X^2) - (z_1^2 + t_{11V}X + t_{12V}X^2)(t_{21\mathbf{h}}x_2 + t_{22\mathbf{h}}x_2^2)$
$f_{31}(X) \leftarrow (z_2^2 + t_{21V}X + t_{22V}X^2)(t_{11u}x_1 + t_{12u}x_1^2) - (z_1^2 + t_{11V}x_1 + t_{12V}x_1^2)(t_{21u}X + t_{22u}X^2)$
$f_{32}(X) \leftarrow (z_2^2 + t_{21V}x_2 + t_{22V}x_2^2)(t_{11u}X + t_{12u}X^2) - (z_1^2 + t_{11V}X + t_{12V}X^2)(t_{21u}x_2 + t_{22u}x_2^2)$
For $i = 1, 2$ do
  $l_i(X) \leftarrow (a_{i\mathbf{g}} - z_i \cdot \mathbf{1}^n) + s_{i\mathbf{g}} \cdot X; \quad r_i(X) \leftarrow \mathbf{y}^n \circ (a_{i\mathbf{h}} + z_i \cdot \mathbf{1}^n + s_{i\mathbf{h}} \cdot X) + z_i^2 \cdot \mathbf{2}^n$
$\delta(y,z) \leftarrow (z - z^2)\langle \mathbf{1}^n, \mathbf{y}^n\rangle - z^3\langle \mathbf{1}^n, \mathbf{2}^n\rangle$
$f_4(X) \leftarrow (z_2^2 + t_{21V}X + t_{22V}X^2)(\delta(y_1,z_1) + t_{11g}x_1 + t_{12g}x_1^2 - \hat{t}_1) - (z_1^2 + t_{11V}x_1 + t_{12V}x_1^2)(\delta(y_2,z_2) + t_{21g}X + t_{22g}X^2$
  $- \langle a_{2\mathbf{g}} + Xs_{2\mathbf{g}} - z_2\mathbf{1}^n, (a_{2\mathbf{h}} + Xs_{2\mathbf{h}} + z_2\mathbf{1}^n) \circ \mathbf{y_2}^n + z_2^2\mathbf{2}^n\rangle)$
$f_5(X) \leftarrow (\langle a_{2\mathbf{g}}, \mathbf{2}^n\rangle)(z_1^2 + t_{11V}X + t_{12V}X^2) + \delta(y_1,z_1) + t_{11g}X + t_{12g}X^2 - \langle l_1(X), r_1(X)\rangle$
Return $\mathsf{SZ}(f_{11}(X), x) \vee \mathsf{SZ}(f_{12}(X), x) \vee \mathsf{SZ}(f_{21}(X), x) \vee \mathsf{SZ}(f_{22}(X), x) \vee \mathsf{SZ}(f_{31}(X), x) \vee \mathsf{SZ}(f_{32}(X), x) \vee \mathsf{SZ}(f_4(X), x) \vee \mathsf{SZ}(f_5(X), x)$

Fig. 33. The functions CheckBad function for the RngPf.

Now, using $(e_{\mathbf{g}}^{(i,2)}, e_{\mathbf{h}}^{(i,2)}, e_u^{(i,2)}, e_g^{(i,2)}, e_h^{(i,2)}) = (\mathbf{0}^n, \mathbf{0}^n, 0, 0, 0)$ for $i = 1, 2$ and that τ_1, τ_2 are not bad, we can conclude like in Theorem 4 (since the values $((e_{\mathbf{g}}^{(2)}, e_{\mathbf{h}}^{(2)}, e_u^{(2)}, e_g^{(2)}, e_h^{(2)})$ and bad challenge function for challenges w and x_i's are defined identically to here) that for $i = 1, 2$

$$\langle a_{i\mathbf{g}} + x_i s_{i\mathbf{g}} - z_i\mathbf{1}^n, (a_{i\mathbf{h}} + x_i s_{i\mathbf{h}} + z_i\mathbf{1}^n) \circ \mathbf{y_i}^n + z_i^2\mathbf{2}^n\rangle = \hat{t}_i \ .$$

Now, using $e_g^{(1)} = 0$, we get that

$$\begin{aligned}(z_2^2 + t_{21V}x_2 + t_{22V}x_2^2)(\delta(y_1,z_1) + t_{11g}x_1 + t_{12g}x_1^2 - \hat{t}_1) = \\ (z_1^2 + t_{11V}x_1 + t_{12V}x_1^2)(\delta(y_2,z_2) + t_{21g}x_2 + t_{22g}x_2^2 \\ - (\langle a_{2\mathbf{g}} + x_2 s_{2\mathbf{g}} - z_2\mathbf{1}^n, (a_{2\mathbf{h}} + x_2 s_{2\mathbf{h}} + z_2\mathbf{1}^n) \circ \mathbf{y_2}^n + z_2^2\mathbf{2}^n\rangle)) \ .\end{aligned}$$

Since (τ_1, τ_2) are not bad we can show that

$$(\delta(y_1,z_1) + t_{11g}x_1 + t_{12g}x_1^2 - \hat{t}_1)(z_1^2 + t_{11V}x_1 + t_{12V}x_1^2)^{-1} = \langle a_{2\mathbf{g}}, \mathbf{2}^n\rangle \ .$$

This means that $v_1^* = \langle a_{2\mathbf{g}}, \mathbf{2}^n\rangle$. Now plugging in the value of $\hat{t}_1$ into above and using (τ_1, τ_2) are not bad we can show that

$$\langle a_{2\mathbf{g}}, \mathbf{2}^n\rangle = \langle a_{1\mathbf{g}}, \mathbf{2}^n\rangle \ , \ a_{1\mathbf{h}} = a_{1\mathbf{g}} - \mathbf{1}^n \ , \ a_{1\mathbf{h}} \circ a_{1\mathbf{g}} = \mathbf{0}^n \ .$$

From this it follows that $\langle a_{2\mathbf{g}}, \mathbf{2}^n \rangle \in [0, 2^n - 1]$ i.e., $v_1^* \in [0, 2^n - 1]$. So we have shown that $v_1^* \in [0, 2^n - 1]$ and $g^{v_1^*} h^{\gamma_1^*} = V$. Hence we proved that for a fixed (τ_1, τ_2), $\mathsf{h}(\tau_1, \tau_2)$ returns a trivial discrete logarithm relation and transcripts (τ_1, τ_2) are not bad then the event E_1 cannot happen. This concludes the proof. □

Acknowledgements.

We thank Joseph Jaeger for extensive discussions and his involvement in the earlier stages of this work. We thank the anonymous reviewers of Eurocrypt 2021 for their feedback on an earlier version of this paper. This work was partially supported by NSF grants CNS-1930117 (CAREER), CNS-1926324, CNS-2026774, a Sloan Research Fellowship, and a JP Morgan Faculty Award.

References

BBB+18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.

BBHR19. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 701–732. Springer, Heidelberg, August 2019.

BCC+16. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357. Springer, Heidelberg, May 2016.

BCR+19. Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Heidelberg, May 2019.

BCS16. Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Heidelberg, October / November 2016.

BFS20. Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 677–706. Springer, Heidelberg, May 2020.

BHT20. Itay Berman, Iftach Haitner, and Eliad Tsfadia. A tight parallel repetition theorem for partially simulatable interactive arguments via smooth KL-divergence. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 544–573. Springer, Heidelberg, August 2020.

BMM+20. Benedict Bünz, Mary Maller, Pratyush Mishra, Nirvan Tyagi, and Psi Vesely. Proofs for inner pairing products and applications. *Cryptology ePrint Archive:2019/1177*, 2020.

BR93. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

CCH+18. Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, and Ron D. Rothblum. Fiat-Shamir from simpler assumptions. Cryptology ePrint Archive, Report 2018/1004, 2018. https://eprint.iacr.org/2018/1004.

CCH+19. Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-Shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *51st ACM STOC*, pages 1082–1090. ACM Press, June 2019.

CHM+20. Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Heidelberg, May 2020.

CL10. Kai-Min Chung and Feng-Hao Liu. Parallel repetition theorems for interactive arguments. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 19–36. Springer, Heidelberg, February 2010.

FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018.

FPS20. Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind schnorr signatures and signed ElGamal encryption in the algebraic group model. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 63–95. Springer, Heidelberg, May 2020.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

GI08. Jens Groth and Yuval Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 379–396. Springer, Heidelberg, April 2008.

GMR85. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985.

Gro16. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.

Hai09. Iftach Haitner. A parallel repetition theorem for any interactive argument. In *50th FOCS*, pages 241–250. IEEE Computer Society Press, October 2009.

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

HPWP10. Johan Håstad, Rafael Pass, Douglas Wikström, and Krzysztof Pietrzak. An efficient parallel repetition theorem. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 1–18. Springer, Heidelberg, February 2010.

JT20. Joseph Jaeger and Stefano Tessaro. Expected-time cryptography: Generic techniques and applications to concrete soundness. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 414–443. Springer, Heidelberg, November 2020.

Lee20. Jonathan Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. *Cryptology ePrint Archive:2020/1274*, 2020.

LFKN90. Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. In *31st FOCS*, pages 2–10. IEEE Computer Society Press, October 1990.

Lin01. Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 171–189. Springer, Heidelberg, August 2001.

Mau05. Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Heidelberg, December 2005.

MBKM19. Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019.

Mon. Monero to become first billion-dollar crypto to implement 'bulletproofs' tech. https://www.coindesk.com/monero-to-become-first-billion-dollar-crypto-to-implement-bulletproofs-tech.

NRSW20. Jonas Nick, Tim Ruffing, Yannick Seurin, and Pieter Wuille. MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. Cryptology ePrint Archive, Report 2020/1057, 2020. https://eprint.iacr.org/2020/1057.

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

WTs+18. Riad S. Wahby, Ioanna Tzialla, abhi shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zkSNARKs without trusted setup. In *2018 IEEE Symposium on Security and Privacy*, pages 926–943. IEEE Computer Society Press, May 2018.

Supplementary Materials

A Concurrent work by Bünz *et. al.*

Bünz *et. al.* [BMM+20] analyse the soundness of the non-interactive protocol obtained by applying the Fiat-Shamir transform to a generalized version of the inner-product argument of Bulletproofs in the AGM. Their analysis is asymptotic and they do not give a concrete bound in the paper.

However, as far as we can tell, making their analysis concrete against an algebraic t-time prover making q random oracle queries would give a bound which contains the term $\sqrt{q \cdot \mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(t)}$. In fact, we think this term may be larger, and of order $q\sqrt{q \cdot \mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(t)} = \sqrt{q^3 \cdot \mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(t)}$, but we are not sure due to a lack of concrete analysis in the paper. This bound is not tight – the multiplicative factor of q before the $\mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(t)$ term is due to reduction to a problem that does not have a tight reduction to the discrete logarithm problem.

This multiplicative factor of q would already be a problem. In the generic-group model, for example, this would result in a term $qt^2/p \approx t^3/p$ (assuming $q \approx t$), which only gives us roughly 85 bits of security on a 256-bit curve.

In our analysis, we give a single reduction to the discrete logarithm relation problem whose hardness is tightly implied by the hardness of discrete logarithm problem, we avoid this multiplicative factor.

Additionally, analysing the inner product argument in the AGM in isolation does not directly give the soundness bound for the full protocol because the bases to which elements are described are not necessarily the same as those that would be available to a cheating algebraic prover against the full protocol.