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$^+$18], which are argument systems with applications across the cryptocurrencies and in verifiably deterministic signatures [NRSW20], which in turn optimize prior work [BCC$^+$16]. The soundness[1] 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]. As these are $\Theta(\log(n))$-round protocols, where n roughly corresponds to the instance size, the (folklore) analysis of the FS transform gives no useful guarantees:[2] 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. For a DLP-based protocol, we certainly 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 interactive arguments based on the hardness of the DLP and related problems, in the *algebraic group model* (AGM) [FKL18]. In the AGM, the adversary provides representations of group elements to the reduction (or to the extractor), and the model has been used already for a number of analyses in the literature. In contrast to prior work [FKL18] on AGM concrete security analysis for linear-PCP based SNARKs [Gro16], which are obtained from two-round protocols, we look at multi-round *public-coin* protocols *and* their non-interactive version obtained via the Fiat-Shamir transform. We aim for bounds with *linear* degradation in the number of random oracle queries, which is essentially tight. (In fact, we will target tightness even with respect to the statement size to be proved.)

The analysis of such protocols is equivalent to analyzing the stronger notion of soundness – *state-restoration soundness* [BCS16,Hol19] – for the 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. State-restoration soundness is *tightly* related to the soundness of the non-interactive argument obtained via the Fiat-Shamir transform. No non-trivial bounds on state-restoration soundness are currently known on any non-constant round *argument*.

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

[2] 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 soundness in the AGM, and apply it to three case studies. (In fact, we target a stronger property of *witness-extended emulation* [Lin01,GI08] that establishes a proof-of-knowledge property.) We give concrete bounds for Bulletproofs, as well as for the Sonic proof system [MBKM19]. Both protocols have previously been analyzed only with respect to plain soundness in the interactive setting, using the forking lemma of Bootle *et al.* [BCC+16], which was only very recently made concrete in [JT20].

We in fact believe that our technique can apply to a number of other protocols based on DLP-variants and that support the AGM, such as Hyrax [WTs+18] or pairing-based instantiations of IOPs [BFS20,CHM+20], and leave their analysis for future work.

We stress that our approach differs formally from recent works (e.g., [MBKM19,CHM+20]) which use the AGM to give a heuristic validation of the security of a *component* of a scheme (e.g., a polynomial commitment scheme), which is then however assumed to satisfy extractability properties compatible with that of 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. (As we point out in our technical overview below, these results actually do not give a full-fledged proof in the AGM for a series of subtle reasons, and modularity is non-obvious in AGM proofs.)

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.[3] For example, in the former, a prover shows in $O(\log n)$ rounds that for a given $C = g^v$ in a cyclic group $\mathbb{G}$ of prime order p we have $v \in [0, 2^n)$. (In fact, this also works as a PoK when v is in a Pedersen's commitment, but we stick with the easier case here, as it allows us to express the quantitative aspects in terms of soundness, as opposed to proof-of-knowledge security.)

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 soundness with probability, roughly, at most

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

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 – and similar bounds also hold on concrete version of proof-of-knowledge security in the AGM.

TIGHTNESS AND DISCUSSION. Given $q < t$, the above bound implies in particular that for most values of n (one should think of $n = 2^{20}$ and $p = 2^{256}$ as representative values), the term $O(qn/p)$ is not leading for groups where generic attacks against the DLP are best possible. 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 tight, given that breaking the DLP would directly give us an attack. This makes our bound essentially exact (up to small constants).

The interesting feature of the bound is that both terms are decoupled (this was not the case in the concrete analysis from [JT20] for the interactive case, where a term $n^3 t/\sqrt{p}$ appears in the generic-group model bound.) Particular care in our proof is needed to ensure that the DLP term

[3] For arithmetic circuit satisfiability (ACS), [BBB+18] gives an argument for a more general relation (a special case of the relation is ACS). We only consider the protocol for ACS, and we in fact do not know whether the more general case admits tight bounds.

is not multiplied by q – as this would 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. Such a multiplicative term q is for example unavoidable if we obtain a bound in the interactive setting first (for plain soundness), and then apply a generic analysis of the FS transform, even if the protocol has just three rounds (i.e., one single challenge).

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 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 two challenges. 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^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 only consider a simpler version of Sonic that omits the signature of correct computation. We believe that our proofs extend to the more efficient variant presented in [MBKM19], but our pedagogical point here is that our framework can improve soundness even for constant-round protocols. Similarly, we ignore the stronger requirement of updatable witness-extended emulation.

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.

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 reduce 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 BCP 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].

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 – unlike prior work [FKL18], our framework in particular allows us to handle protocol inputs which are also group elements.

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]. For example, consider Schnorr's protocol [Sch90] for a cyclic group $\mathbb{G} = \langle g \rangle$ with prime order $|\mathbb{G}| = p$, which, given an input $X \in \mathbb{G}$ proves knowledge of a witness w such that $g^w = X$. In this protocol, the prover sends $A \leftarrow g^a$ for a random $a \leftarrow\!\!\$\, \mathbb{Z}_p$, the verifier responds with a random challenge $c \leftarrow\!\!\$\, \mathbb{Z}_p$, and the prover finally sends $d = cw + a$ to the verifier, which accepts if and only if $A \cdot X^c = g^d$. In the standard model, security follows from *special soundness* – the fact that given two accepting transcripts $\tau = (A, c, d)$ and $\tau' = (A, c', d')$ with $c \neq c'$ we know that $w = (d - d')/(c - c')$. Given a prover succeeding with probability ε, we obtain an algorithm computing w from X with probability ε^2 by the Forking Lemma [PS00].

In contrast, in the AGM, from a single accepting transcript $\tau = (A, c, d)$, the extractor learns additionally $a_X, a_g \in \mathbb{Z}_p$ such that $A = X^{a_X} \cdot g^{a_g}$. Therefore,

$$g^{d - a_G} = X^{a_X + c} ,$$

and thus, unless $c + a_X = 0$ (which would have happened with probability $1/p$ after fixing a_X), the extractor can output $w = (d - a_G)/(a_X + c)$. (This fact was recently exploited in [FPS20] to show tight bounds for Schnorr Signatures in the AGM.)

We also note that the AGM is not a panacea and does not trivialize the problem. In fact, there are protocols which do not allow for online extraction. We give an example in Appendix A. This makes the question of whether AGM online extraction is possible very subtle.

A GENERAL FRAMEWORK. The above discussion refers to conventional provers, which have a single interaction with a verifier. 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 choices of challenges may prevent witness extraction – in Schnorr's protocol, this is exactly the case when $c + a_X = 0$.
- **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.

The above example with Schnorr's protocol only encounters the first situation – indeed, it is a *proof* of knowledge (as opposed to an *argument* of knowledge). However, we give a more involved example next, which considers a simplified instance of the inner product argument at the core of Bulletproofs.

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 $\langle \mathbf{a}, \mathbf{b} \rangle$.[4] 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

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

[4] 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 $\prod_{i=1}^{n} g_i^{a_i}$.

The prover sends $a' = a_1x + a_2x^{-1}$ and $b' = b_1x^{-1} + b_2x$ to the verifier, which in turns accepts if and only if

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

EXRACTION FOR $n = 2$. To see how extraction works, let

$$\tau = ((L, R), x, (a', b'))$$

be an accepting transcript for the protocol, i.e., $P' = (g')^{a'}(h')^{b'}u^{a'b'}$. Now, in the AGM, the transcript contains the representations of the group elements $L = g_1^{l_{g_1}}g_2^{l_{g_2}}h_1^{l_{h_1}}h_2^{l_{h_2}}u^{l_u}P^{l_P}$ and $R = g_1^{r_{g_1}}g_2^{r_{g_2}}h_1^{r_{h_1}}h_2^{r_{h_2}}u^{r_u}P^{r_P}$, which we can use to find $e_{g_1}, e_{g_2}, e_{h_1}, e_{h_2}, e_P, e_u$ such that

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

For example $e_{g_1} = x^{-1}a' - l_{g_1}x^2 - r_{g_1}x^{-2}$ and $e_P = 1 + l_Px^2 + r_Px^{-2}$. If $e_P \neq 0$ (which is true with high probability over the choice of x), an extraction procedure can simply return $\mathbf{a}' = (e_{g_1}/e_P, e_{g_2}/e_P)$ and $\mathbf{b}' = (e_{h_1}/e_P, e_{h_2}/e_P)$ as the witness. At first, it looks like we are done – however, we must additionally verify that $\langle \mathbf{a}', \mathbf{b}' \rangle = e_u/e_P$, for otherwise we failed to find a valid witness.

We will prove that if this is not true, then we can solve the discrete logarithm problem in the group $\mathbb{G}$. To this end, we construct 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.) 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

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

It is important here that the representation of P is available to the *reduction*, i.e., to $\mathcal{A}$, but was not to the extractor. Then, $\mathcal{A}$ simulates the verifier to $\mathcal{P}$ – if an accepting transcript is generated, we get values $e_{g_1}, e_{g_2}, e_{h_1}, e_{h_2}, e_u, e_P \in \mathbb{Z}_p$ as above, but because we *know* the representation of P, $\mathcal{A}$ can actually find $e'_{g_1}, e'_{g_2}, e'_{h_1}, e'_{h_2}, e'_u \in \mathbb{Z}_p$ such that

$$g_1^{e'_{g_1}}g_2^{e'_{g_2}}h_1^{e'_{h_1}}h_2^{e'_{h_2}}u^{e'_u} = 1 . \tag{3}$$

For example, $e'_{g_1} = e_{g_1} - e_Pp_{g_1}$, and the other values can be derived analogously. So, what we need to prove is that if $(e'_{g_1}, e'_{g_2}, e'_{h_1}, e'_{h_2}, e'_u)$ are all zero, then $\langle \mathbf{a}', \mathbf{b}' \rangle = e_u/e_P$, which is equivalent to saying that if the latter condition does not hold, then $\mathcal{A}$ has found a non-trivial relation. Note that these values being all 0 implies in particular that

$$e_{g_1} = e_Pp_{g_1} \ , \ \ e_{g_2} = e_Pp_{g_2} \ , \ \ e_{h_1} = e_Pp_{h_1} \ , \ \ e_{h_2} = e_Pp_{h_2} \ , \ \ e_u = e_Pp_u \ .$$

Therefore,

$$\langle \mathbf{a}', \mathbf{b}' \rangle = p_{g_1}p_{h_1} + p_{g_2}p_{h_2} \ .$$

Hence, the goal is to show that $p_{g_1}p_{h_1} + p_{g_2}p_{h_2} = p_u = e_u/e_P$.

Assuming that $x \neq 0$, plugging in the values of e_{g_1}, e_P from above into $e_{g_1} = e_Pp_{g_1}$, and solving for a', gives us

$$a' = x^3(l_{g_1} + l_Pp_{g_1}) + xp_{g_1} + x^{-1}(r_{g_1} + r_Pp_{g_1}) \ .$$

We get another such equation via $e_{g_2} = e_P p_{g_2}$,

$$a' = x(l_{g_2} + l_P p_{g_2}) + x^{-1} p_{g_2} + x^{-3}(r_{g_2} + r_P p_{g_2}) \ .$$

With high probability over the choice of x's, by the Schwartz-Zippel Lemma, we can infer by equating both right-hand sides that

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

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

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

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

$$a'b' = x^2(l_u + l_p P_u) + p_u + x^{-2}(r_u + r_P p_u) \ .$$

But by the above,

$$a'b' = p_{g_1} p_{h_1} + p_{g_2} p_{h_2} + p_{g_1} p_{h_2} x^2 + p_{g_2} p_{h_1} x^{-2} \ .$$

Therefore, again by equating the right-hand sides, and the Schwartz-Zippel Lemma, we must have $p_u = p_{g_1} p_{h_1} + p_{g_2} p_{h_2}$, as we wanted to show.

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 the core 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 $[N] = \{1, \ldots, N\}$. The symbol $\varnothing$ denotes the empty set. The cardinality of a set S is denoted by $|S|$. Sampling c uniformly at random from a set S is denoted by $c \leftarrow\$\, S$. We let $y \leftarrow\$\, \mathcal{A}^{O}(x_1, x_2, \ldots,)$ denote the execution of a non-deterministic algorithm $\mathcal{A}$ on input $x_1, x_2, \ldots$ that has oracle access to O. We use $\wedge, \vee$ for logical operators "and", "or" respectively. We use $\Pr[\mathsf{G}]$ 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.,

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

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 inner product between two vector polynomials is defined as

$$\langle l(X), r(X) \rangle = \sum_{i=0}^{d} \sum_{j=0}^{i} \langle \mathbf{l}_i, \mathbf{r}_j \rangle X^{i+j} \ .$$

Note that evaluating two vector polynomials at some point x and taking their inner product gives the same result as taking their inner product and then evaluating the resulting polynomial at x.

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. All logarithms in this paper have base 2.

SCHWARTZ-ZIPPEL LEMMA. Let $f(X_1, \ldots, X_n)$ be a n variate polynomial. We use $f(x_1, \ldots, x_n)$ to denote the evaluation of f at the point $(x_1, \ldots, x_n)$ throughout the paper. 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. Then $\Pr[f(x_1, \ldots, x_n) = 0] \leqslant \frac{d}{|\mathbb{F}|}$, where the probability is over the choice of $x_1, \ldots, x_n$ according to $x_i \leftarrow_\$ \mathbb{F}$.*

THE DISCRETE LOGARITHM PROBLEM. The game $\mathsf{G}_{\mathbb{G}}^{\mathsf{dl}}$ 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

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

```
Game $\mathsf{G}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{A}, \lambda)$:
    $g \leftarrow_{\$} \mathbb{G}_\lambda{}^*$; $h \leftarrow_{\$} \mathbb{G}_\lambda$
    $a \leftarrow_{\$} \mathcal{A}_\lambda(g, g^v)$
    Return $(g^a = h)$
```

```
Game $\mathsf{G}^{\mathsf{dl\text{-}rel}}_{\mathbb{G},n}(\mathcal{A}, \lambda)$:
    $g_1, \ldots, g_n \leftarrow_{\$} \mathbb{G}_\lambda$
    $(a_1, \ldots, a_n) \leftarrow_{\$} \mathcal{A}_\lambda(g_1, \ldots, g_n)$
    Return $(\prod_{i=1}^{n} g_i^{a_i} = 1 \wedge (a_1, \ldots, a_n) \neq \mathbf{0}^n)$
```

```
Game $\mathsf{G}^{q\text{-}\mathsf{dl}}_{\mathbb{G}}(\mathcal{A}, \lambda)$:
    $g \leftarrow_{\$} \mathbb{G}_\lambda{}^*$; $x \leftarrow_{\$} \mathbb{Z}_{p(\lambda)}$
    $x' \leftarrow_{\$} \mathcal{A}_\lambda(\{g^x\}_{x=-q}^{q})$
    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 $\mathbb{G}_\lambda$.

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

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

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.

THE q-DLOG PROBLEM. The game $\mathsf{G}^{q\text{-}\mathsf{dl}}_{\mathbb{G}}$ 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

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

3 Interactive Proofs and Arguments

3.1 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] is a triple $\mathsf{IP} = (\mathsf{IP.Setup}, \mathsf{IP.P}, \mathsf{IP.V})$ of algorithms: (1) the *setup algorithm* $\mathsf{IP.Setup}$ which generates the public parameters pp, (2) the *prover* $\mathsf{IP.P}$ and (3) the *verifier* $\mathsf{IP.V}$. 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 $\mathsf{IP.V}$ to $\mathsf{IP.P}$ are fresh random values from some understood set (which we refer to as *challenges*).

```
Game $\mathsf{SRS}^{\mathcal{P}}_{\mathsf{IP}}(\lambda)$:
$\mathsf{win} \leftarrow \mathtt{false}; \mathsf{tr} \leftarrow \varepsilon$
$\mathsf{pp} \leftarrow_{\$} \mathsf{IP.Setup}(1^\lambda)$
$(x, \mathsf{st}_{\mathcal{P}}) \leftarrow_{\$} \mathcal{P}_\lambda(\mathsf{pp})$
Run $\mathcal{P}_\lambda^{\mathbf{O}_{\text{ext}}}(\mathsf{st}_{\mathcal{P}})$
Return win
```

```
Oracle $\mathbf{O}_{\text{ext}}(\tau = (a_1, c_1, \ldots, a_{i-1}, c_{i-1}), a_i)$:
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
    $d \leftarrow \mathsf{IP.V}(\mathsf{pp}, x, (\tau, a_i))$; $\mathsf{tr} \leftarrow \mathsf{tr} \,\|\, (\tau, a_i)$
    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.

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 `true` if $(\mathsf{pp}, x, w) \in R$ and `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 *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 $\mathbf{O}_{\text{ext}}$, 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 $\mathbf{O}_{\text{ext}}$ 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*, $\mathsf{Adv}^{\mathsf{srs}}_{\mathsf{IP}}(\mathcal{P}, \lambda) = \Pr\left[\mathsf{SRS}^{\mathsf{IP}}_{\mathcal{P}}(\lambda)\right]$. 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.

3.2 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 $\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}]$. Suppose the length of the i^{th} challenge in IP is cLen_i. 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 a transcript $\tau = (a_1, c_1, a_2, c_2, \ldots, a_r, c_r, a_{r+1})$, 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 $\mathsf{IP.V}$ returns 1 on input (pp, x, τ).

We use the game $\mathsf{FSRO}_{\mathsf{IP}}$ in Figure 3 to formally capture the soundness of the non-interactive argument $\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}]$ against a non-uniform prover $\mathcal{P}$. For security parameter λ, the advantage of $\mathcal{P}$ against the soundness of $\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}]$ is $\mathsf{Adv}^{\mathsf{snd}}_{\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}]}(\mathcal{P}, \lambda) = \Pr\left[\mathsf{FSRO}^{\mathcal{P}}_{\mathsf{IP}}(\lambda)\right]$.

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

Theorem 1. *Let* IP *be a* $r = r(\lambda)$*-challenge public coin interactive protocol 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* $i \in \{1, \ldots, r\}$*. For every non-uniform cheating prover* $\mathcal{P}^*$ *making* $q = q(\lambda)$ *random oracle queries, there exists a non-uniform prover* $\mathcal{P}$ *such that for all* $\lambda \in \mathbb{N}^+$

$$\mathsf{Adv}^{\mathsf{snd}}_{\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}]}(\mathcal{P}^*, \lambda) \leqslant \mathsf{Adv}^{\mathsf{srs}}_{\mathsf{IP}}(\mathcal{P}, \lambda) + \frac{q+1}{2^{\mathsf{sLen}(\lambda)}} \ .$$

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

```
Game $\mathsf{FSRO}^{\mathcal{P}}_{\mathsf{IP}}(\lambda)$:
$\mathsf{pp} \leftarrow \mathsf{IP.Setup}(1^\lambda)$; $(x, \mathsf{st}_\mathcal{P}) \leftarrow_\$ \mathcal{P}_\lambda(\mathsf{pp})$; $H \leftarrow_\$ \Omega_{\mathsf{hLen}(\lambda)}$
$(a_1, c_1, a_2, c_2 \ldots, a_r, c_r, a_{r+1}) \leftarrow_\$ \mathcal{P}^H_\lambda(\mathsf{st}_\mathcal{P})$
For $\exists i$: $c_i \neq H(\mathsf{pp}, x, a_1, c_1, \ldots, a_{i-1}, c_{i-1}, a_i)[: \mathsf{cLen}_i(\lambda)]$ then return false
Return $\mathsf{IP.V}(\mathsf{pp}, x, (a_1, c_1, \ldots, a_r, c_r, a_{r+1}))$
```

Fig. 3. Soundness for the Fiat-Shamir transform. This game defines the advantage a non-uniform prover $\mathcal{P}$ against the soundness of $\mathsf{FSRO}[\mathsf{IP}]$. Here, IP has $r = r(\lambda)$ challenges where the i^{th} challenge is of length $\mathsf{cLen}_i(\lambda)$ such that $\mathsf{sLen}(\lambda) \leqslant \mathsf{cLen}_i(\lambda) \leqslant \mathsf{hLen}(\lambda)$. The set $\Omega_{\mathsf{hLen}(\lambda)}$ contains all hash functions mapping $\{0,1\}^*$ to $\{0,1\}^{\mathsf{hLen}(\lambda)}$.

Proof. We shall construct a prover $\mathcal{P}$ playing $\mathsf{SRS}_{\mathsf{IP}}$ that runs the cheating prover $\mathcal{P}^*$ and simulates the game $\mathsf{FSRO}_{\mathsf{IP}}$ to it.

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 $i \in \{1, \ldots, r+1\}$. Without loss of generality we assume that $\mathcal{P}^*$ does not repeat any queries to the random oracle.

The first stage of $\mathcal{P}$ on input pp shall run the first stage of the $\mathcal{P}^*$ on pp. If $\mathcal{P}^*$ returns $(x, \mathsf{st}_{\mathcal{P}*})$, $\mathcal{P}$ returns $(x, \mathsf{st}_\mathcal{P} = (\mathsf{st}_{\mathcal{P}*}, \mathsf{pp}, x))$. The second stage of $\mathcal{P}$ 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 $\mathcal{P}^*$ on $\mathsf{st}_{\mathcal{P}*}$. It simulates the random oracle H to $\mathcal{P}^*$ as follows. On receiving a H query on y

1. $\mathcal{P}$ 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, $\mathcal{P}$ returns a randomly sampled string from $\{0,1\}^{\mathsf{hLen}}$. If the check succeeds, $\mathcal{P}$ chooses the longest such state s.
2. $\mathcal{P}$ parses as y as (pp, x, s, t) and makes a query to $\mathbf{O}_{\text{ext}}$ on (s, t) ans receives c as the response. $\mathcal{P}$ 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 $\mathcal{P}^*$ returns an output τ, $\mathcal{P}$ queries $\mathbf{O}_{\text{ext}}$ on τ and stops. It follows that $\mathcal{P}$ makes no more than q queries to its oracle and is nearly as efficient as $\mathcal{P}^*$.

Suppose the game $\mathsf{FSRO}^{\mathcal{P}^*}_{\mathsf{IP}}$ returns true. In other words $\mathcal{P}^*$ returns an accepting proof, i.e., it returns $\tau = (a_1, c_1, \ldots, a_r, c_r, a_{r+1})$ 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 $\mathsf{IP.V}(\mathsf{pp}, x, \tau)$ returns true.

Let $\tau_i = (a_1, c_1, \ldots, a_{i-1}, c_{i-1}, a_i)$. Now, let E be the event that $\mathcal{P}^*$ 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, $\mathcal{P}$ must have queried $\mathbf{O}_{\text{ext}}$ on $\tau_1, \tau_2, \ldots, \tau_r$ in order and finally queried τ. Since these queries to $\mathbf{O}_{\text{ext}}$ were in this order, $\mathbf{O}_{\text{ext}}$ must have set win to true when finally queried on τ. Therefore, when the game $\mathsf{FSRO}^{\mathcal{P}^*}_{\mathsf{IP}}$ returns true and E happens, the game $\mathsf{SRS}^{\mathcal{P}}_{\mathsf{IP}}$ returns true.

Hence, we need to upper bound the probability that τ is an accepting transcript and the event E does not happen, in order to upper bound $\mathsf{Adv}^{\mathsf{snd}}_{\mathsf{FS^{RO}[IP]}}(\mathcal{P}^*, \lambda)$ in terms of $\mathsf{Adv}^{\mathsf{srs}}_{\mathsf{IP}}(\mathcal{P}, \lambda)$.

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 $\mathcal{P}^*$ 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 $1/2^{\mathsf{sLen}}$ 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 $\mathcal{P}^*$ 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

$$\mathsf{Adv}^{\mathsf{snd}}_{\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}]}(\mathcal{P}^*, \lambda) \leqslant \mathsf{Adv}^{\mathsf{srs}}_{\mathsf{IP}}(\mathcal{P}, \lambda) + \frac{q+1}{2^{\mathsf{sLen}(\lambda)}} \ .$$

□

Here 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 $(q + 1)/(p - 1)$.

ZERO-KNOWLEDGE. An interactive protocol IP for a relation R is said to be honest verifier zero-knowledge (HVZK) if there exists an efficient simulator $\mathcal{S}$ such that for all $(\mathsf{pp}, x, w) \in R$, the output of $\mathcal{S}(\mathsf{pp}, x)$ is indistinguishable from the view of IP.V that has as input (pp, x) and is interacting with IP.P with input (pp, x, w). The view of IP.V consists of the transcript of the interaction and its randomness.

If a public coin protocol IP is HVZK, applying the Fiat-Shamir transform to it produces a non-interactive zero-knowledge argument. The interactive protocols we consider in this paper (Bulletproofs, Sonic) are HVZK.

4 Arguments of Knowledge in the AGM

This paper focuses on arguments of knowledge based on a (cyclic) group, for which we prove concrete security in the *algebraic group model* (AGM) [FKL18]. More specifically, the security property we target is an extension of *witness-extended emulation* [Lin01,GI08] to consider state-restoration provers. Via Theorem 2, this will give us tight proof of knowledge bounds (in the random-oracle model) for the NIZK argument obtained by applying the Fiat-Shamir transform to these arguments. This section will develop in particular a definition for the specific case of online extraction in the AGM. (A more general definition could be given, but this is sufficient for our purposes.) We also provide a framework that we will adopt to study concrete protocols below.

4.1 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, 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.

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

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

Oracle $\mathbf{O}^0_{\mathrm{ext}}(\tau, a)$:
If $\tau \in \mathsf{tr}$ then
  $(\mathsf{resp}, \mathsf{st}_\mathcal{E}) \leftarrow\!\!\$\ \mathcal{E}(\mathsf{st}_\mathcal{E}, [(\tau, a)])$
  $\mathsf{tr} \leftarrow \mathsf{tr} \,\|\, (\tau, a, \mathsf{resp})$
  Return $\mathsf{resp}$
Return $\perp$
```

Fig. 4. Definition of online srs-wee security in the AGM. The games WEE-1, WEE-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.

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.) This requires a little care, as 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. The extractor should not learn too much in order not to trivialize its task. It is also important to note that a group-based interactive proof does not necessarily allow for AGM online extraction – we elaborate on this in Appendix A.

ONLINE SRS-WEE SECURITY. The definition consists of two games – denoted $\mathsf{WEE}\text{-}1_{\mathsf{IP}}^{\mathcal{P}_{\mathsf{alg}},\mathcal{D}}$ and $\mathsf{WEE}\text{-}0_{\mathsf{IP},R}^{\mathcal{E},\mathcal{P}_{\mathsf{alg}},\mathcal{D}}$, and described in Figure 4. 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 and in the input x.) 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 `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 Acc(tr).

```
Game $\mathsf{FS\text{-}EXT}^{\mathcal{P}_{\mathsf{alg}},\mathcal{E}}_{\mathsf{IP},R}(\lambda)$:
$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{IP.Setup}(1^\lambda)$; $(x, \mathsf{st}_{\mathcal{P}}) \leftarrow\!\!\$\ \mathcal{P}_{\mathsf{alg},\lambda}(\mathsf{pp})$
$\mathsf{st}_{\mathcal{E}} \leftarrow (1^\lambda, \mathsf{pp}, x)$; $\tau \leftarrow \mathcal{P}^{\mathbf{RO}_{\mathrm{ext}}}_{\mathsf{alg},\lambda}(\mathsf{st}_{\mathcal{P}})$; $w \leftarrow\!\!\$\ \mathcal{E}(\mathsf{st}_{\mathcal{E}}, [\tau])$
Return $(\mathsf{IP.V}(\mathsf{pp}, x, \tau) = 1 \wedge (\mathsf{pp}, x, w) \notin R)$

Oracle $\mathbf{RO}_{\mathrm{ext}}(y)$:
$(\mathsf{resp}, \mathsf{st}_{\mathcal{E}}) \leftarrow\!\!\$\ \mathcal{E}(\mathsf{st}_{\mathcal{E}}, [y])$
Return resp
```

Fig. 5. Definition of fs-ext security in the AGM. The game FS-EXT defines fs-ext 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.

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

$$\mathsf{Adv}^{\mathsf{sr\text{-}wee}}_{\mathsf{IP},R}(\mathcal{P}_{\mathsf{alg}}, \mathcal{D}, \mathcal{E}, \lambda) = \Pr\left[\mathsf{WEE\text{-}1}^{\mathcal{P}_{\mathsf{alg}},\mathcal{D}}_{\mathsf{IP}}(\lambda)\right] - \Pr\left[\mathsf{WEE\text{-}0}^{\mathcal{E},\mathcal{P}_{\mathsf{alg}},\mathcal{D}}_{\mathsf{IP},R}(\lambda)\right] . \tag{4}$$

FS-EXT 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. It is defined by the game $\mathsf{FS\text{-}EXT}^{\mathcal{P}_{\mathsf{alg}},\mathcal{E}}_{\mathsf{IP},R}$ in Figure 5. The game is similar to FSRO with the main difference being that the extractor $\mathcal{E}$ simulates the random oracle to the prover $\mathcal{P}_{\mathsf{alg}}$ and outputs a witness after the prover has output a proof. The extractor exploits the fact that $\mathcal{P}_{\mathsf{alg}}$ is algebraic by learning the representation of every input to the oracle $\mathbf{RO}_{\mathrm{ext}}$. Here, the final output of the game is `true` if $\mathcal{P}_{\mathsf{alg}}$ outputs an accepting proof τ (i.e., $\mathsf{IP.V}(\mathsf{pp}, x, \tau)$ returns 1) but the witness output by the extractor is not a valid one. For an interactive proof IP and an associated relation R, algebraic prover $\mathcal{P}_{\mathsf{alg}}$, and an extractor $\mathcal{E}$, we define $\mathsf{Adv}^{\mathsf{fs\text{-}ext}}_{\mathsf{FS}^{\mathbf{RO}}[\mathsf{IP}],R}(\mathcal{P}_{\mathsf{alg}}, \mathcal{E}, \lambda) = \Pr\left[\mathsf{FS\text{-}EXT}^{\mathcal{P}_{\mathsf{alg}},\mathcal{E}}_{\mathsf{IP},R}(\lambda)\right]$.

The following theorem connects the online srs-wee of a public-coin protocol IP and the fs-ext 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 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 $\lambda \in \mathbb{N}^+$*

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

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 (Sketch). This proof is very similar to the proof of Theorem 1, so we just provide a proof sketch.

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 $\mathcal{E}^*$ that simulates the game WEE-0 to $\mathcal{E}$. Suppose $\mathcal{E}^*$ has initial state $(1^\lambda, \mathsf{pp}, x)$. It initializes $\mathsf{st}_{\mathcal{E}}$ to $(1^\lambda, \mathsf{pp}, x)$. It maintains a set of states $\mathcal{S}$ which is initialized to $\{\varepsilon\}$ where ε is the empty string. For every query y that it receives, it checks if for some state $s \in \mathcal{S}$ of length i, there exists $y = (\mathsf{pp}, x, s, t)$ where t is l_{i+1} bits long. If the check fails, it returns a random string in $\{0,1\}^{\mathsf{hLen}(\lambda)}$. Otherwise, it chooses the longest such s and queries (s, t) to $\mathcal{E}$, receives a string c. It samples c' uniformly at random from $\{0,1\}^{\mathsf{hLen}-\mathsf{cLen}_{i+1}}$ and returns (c, c') (we omit the state book-keeping here). When invoked on τ, it runs $\mathcal{E}$ on $(\mathsf{st}_{\mathcal{E}}, \tau)$ and then $(\mathsf{st}_{\mathcal{E}}, \perp)$ and returns whatever $\mathcal{E}$ returns. It follows that $\mathcal{E}^*$ is nearly as efficient as $\mathcal{E}$.

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 WEE-0, Acc returns `true` and $\mathcal{E}$ fails to return a valid witness.

We define adversary $\mathcal{P}_{\mathsf{alg}}$ that runs adversary $\mathcal{P}^*_{\mathsf{alg}}$ in the same way as $\mathcal{P}$ ran $\mathcal{P}^*$ in the proof of Theorem 1. It follows that $\mathcal{P}_{\mathsf{alg}}$ makes at most q queries to its oracle and is nearly as efficient as $\mathcal{P}^*_{\mathsf{alg}}$.

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

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

Since $\mathbf{RO}_{\mathrm{ext}}$ queries are always chosen uniformly at random from $\{0,1\}^{\mathsf{hLen}(\lambda)}$, and no queries are repeated, one can think of it as a random oracle H with image $\{0,1\}^{\mathsf{hLen}(\lambda)}$. So the event E is same as in the proof of Theorem 1, where we upper bounded the probability that $\mathcal{P}^*_{\mathsf{alg}}$ produces an accepting proof and E does not happen to $(q+1)/2^{\mathsf{sLen}(\lambda)}$.

Therefore it follows that

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

□

Here 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 $(q+1)/(p-1)$.

SOUNDNESS FROM PoK. The definition of state-restoration soundness from Section 3.1 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 $\lambda \in \mathbb{N}^+$*

$$\mathsf{Adv}^{\mathsf{srs}}_{\mathsf{IP}}(\mathcal{P}_{\mathsf{alg}}, \lambda) \leqslant \mathsf{Adv}^{\mathsf{sr\text{-}wee}}_{\mathsf{IP},R}(\mathcal{P}_{\mathsf{alg}}, \mathcal{D}, \mathcal{E}, \lambda) .$$

4.2 The Basic Framework

We develop a general framework that we will use, via Theorem 3, 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 $[\tau] \in \mathcal{T}^{\mathsf{IP}}_{\mathsf{Acc}}$. We need to however be a little careful, since the extractor cannot leverage a representation of the group elements in x, as this trivializes extraction. To this end, for an extended transcript $[\tau]$, we write $\overline{[\tau]} = (\mathsf{pp}, x, [a_1], c_1, [a_2], c_2, \ldots, [a_r], c_r, [a_{r+1}])$, i.e., $\overline{[\tau]}$ omits the representation of the input x. For a particular function e we now define the set of extended transcripts on which it succeeds in extracting a valid witness, i.e.,

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

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 $[\tau] \in \mathcal{T}^{\mathsf{IP}}_{\mathsf{Acc}} \backslash \mathcal{T}^{\mathsf{IP},\mathsf{e},R}_{\mathsf{correct}}$. 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 $\mathcal{T}^{\mathsf{IP}}_{\mathsf{Acc}} \backslash \mathcal{T}^{\mathsf{IP},\mathsf{e},R}_{\mathsf{correct}}$ 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

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

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 $\mathcal{T}^{\mathsf{IP}}_{\mathsf{BadCh}}$ 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 $|\mathsf{BadCh}(\tau')| / |\mathsf{Ch}_i| \leqslant \varepsilon$.

The only case that the extractor can now fail is if the execution tree contains an extended transcript $[\tau]$ in the set

$$\mathcal{T}^{\mathsf{IP},\mathsf{e},R}_{\mathsf{fail}} = \mathcal{T}^{\mathsf{IP}}_{\mathsf{Acc}} \setminus (\mathcal{T}^{\mathsf{IP},\mathsf{e},R}_{\mathsf{correct}} \cup \mathcal{T}^{\mathsf{IP}}_{\mathsf{BadCh}}) \ . \tag{5}$$

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 3 (Master Theorem). *Let* IP *be an $r = r(\lambda)$-challenge public coin interactive proof for a relation R. Assume that* BadCh *and* e *are given above. Further, assume that for an $i \in \{1, \ldots, r\}$, 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 $\mathcal{E}$ 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 $\lambda \in \mathbb{N}^+$

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

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.

Proof. The extractor $\mathcal{E}$, as stated above, 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 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 $\overline{\lceil \tau \rceil}$ 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 above).

By definition, $\tau \in \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{IP}}$ 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 $\tau \in \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{IP}}$ is at most $q \cdot \varepsilon$.

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

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

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

□

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 Bulletproof inner product argument (InPrd) in Section 5.1

$$
\begin{array}{lcl}
\mathsf{InPrd.P}(((n,\mathbf{g},\mathbf{h},u),P),(\mathbf{a},\mathbf{b})) & & \mathsf{InPrd.V}((n,\mathbf{g},\mathbf{h},u),P) \\
\mathbf{g}^{(0)} \leftarrow \mathbf{g}; \mathbf{h}^{(0)} \leftarrow \mathbf{h} & & \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} & & n_0 \leftarrow n; P^{(0)} \leftarrow P \\
\text{For } i = 1, \ldots, \log n & & \text{For } i = 1, \ldots, \log n \\
\quad n_i \leftarrow n_{i-1}/2 & & \quad n_i \leftarrow n_{i-1}/2 \\
\quad c_L \leftarrow \langle \mathbf{a}[:n_i], \mathbf{b}[n_i:] \rangle & & \\
\quad c_R \leftarrow \langle \mathbf{a}[n_i:], \mathbf{b}[:n_i] \rangle & & \\
\quad L_i \leftarrow \left(\mathbf{g}^{(i-1)}_{[n_i:]}\right)^{\mathbf{a}[:n_i]} \left(\mathbf{h}^{(i-1)}_{[:n_i]}\right)^{\mathbf{b}[n_i:]} u^{c_L} & & \\
\quad R_i \leftarrow \left(\mathbf{g}^{(i-1)}_{[:n_i]}\right)^{\mathbf{a}[n_i:]} \left(\mathbf{h}^{(i-1)}_{[n_i:]}\right)^{\mathbf{b}[:n_i]} u^{c_R} & \xrightarrow{L_i, R_i} & \\
 & \xleftarrow{x_i} & \quad x_i \leftarrow_{\$} \mathbb{Z}_p^* \\
\quad \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} & & \quad \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} \\
\quad \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}} & & \quad \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}} \\
\quad P^{(i)} \leftarrow L_i^{x_i^2} P^{(i-1)} R_i^{x_i^{-2}} & & \quad P^{(i)} \leftarrow L_i^{x_i^2} P^{(i-1)} R_i^{x_i^{-2}} \\
\quad \mathbf{a}^{(i)} \leftarrow \mathbf{a}^{(i-1)}[:n_i]x^{-1} + \mathbf{a}^{(i)}[n_i:]x & & \\
\quad \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)} & & 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} & \text{Return } (P^{(\log n)} = g^a h^b u^{ab})
\end{array}
$$

Fig. 6. Bulletproofs inner-product argument InPrd.

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 InPrd $=$ 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. It is a $\log n$ round recursive protocol where in every round the prover and the verifier engage in InPrd with vectors of dimension half of that in the previous round.

5.2 Online srs-wee Security of RngPf

We shall assume that RngPf $=$ 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

$$R = \left\{ \Big((n \in \mathbb{N}, g, h \in \mathbb{G}), V \in \mathbb{G}, (v, \gamma \in \mathbb{Z}_p) \Big) : g^v h^\gamma = V \wedge v \in [0, 2^n - 1] \right\} . \qquad (6)$$

Description of RngPf. The RngPf.Setup procedure on input 1^λ returns a positive integer n, $\mathbf{g} \in \mathbb{G}^n, \mathbf{h} \in \mathbb{G}^n, g, h, u \in \mathbb{G}$ where $\mathbf{g}, \mathbf{h}$ are vectors of independent generators and g, h, u are other independent generators of the group $\mathbb{G}$. The instance for RngPf is $V \in \mathbb{G}$ such that an honest prover knows a witness (v, γ) that satisfies $V = g^v h^\gamma$ and $v \in [0, 2^n - 1]$.

```
$\mathsf{RngPf.P}(((n, \mathbf{g}, \mathbf{h}, g, h, u), V), (v, \gamma))$          $\mathsf{RngPf.V}((n, \mathbf{g}, \mathbf{h}, g, h, u), V)$
$\mathbf{a}_L \leftarrow \mathsf{BinRep}(v, n); \mathbf{a}_R \leftarrow \mathbf{a}_L - \mathbf{1}^n$
$\alpha \leftarrow_{\$} \mathbb{Z}_p; A \leftarrow h^{\alpha}\mathbf{g}^{\mathbf{a}_L}\mathbf{h}^{\mathbf{a}_R}$
$\mathbf{s}_L \leftarrow_{\$} \mathbb{Z}_p^n; \mathbf{s}_R \leftarrow_{\$} \mathbb{Z}_p^n$
$\rho \leftarrow_{\$} \mathbb{Z}_p; S \leftarrow h^{\rho}\mathbf{g}^{\mathbf{s}_L}\mathbf{h}^{\mathbf{s}_R}$          $\xrightarrow{A,S}$
          $\xleftarrow{y,z}$ $y, z \leftarrow_{\$} \mathbb{Z}_p^*$
$l(X) \leftarrow (\mathbf{a}_L - z \cdot \mathbf{1}^n) + \mathbf{s}_L \cdot X$          $\delta(y, z) \leftarrow (z - z^2) \cdot \langle \mathbf{1}^n, \mathbf{y}^n \rangle + z^2 \cdot \mathbf{2}^n$
$r(X) \leftarrow \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n + \mathbf{s}_R \cdot X) + z^2 \cdot \mathbf{2}^n$
$t(X) \leftarrow \langle l(X), r(X) \rangle = t_0 + t_1 X + t_2 X^2$
$\tau_1, \tau_2 \leftarrow_{\$} \mathbb{Z}_p$
$T_i \leftarrow g^{t_i} h^{\tau_i}$ for $i \in \{1, 2\}$          $\xrightarrow{T_1,T_2}$
          $\xleftarrow{x}$ $x \leftarrow_{\$} \mathbb{Z}_p^*$
$\mathbf{l} \leftarrow l(x); \mathbf{r} \leftarrow r(x); \hat{t} \leftarrow \langle \mathbf{l}, \mathbf{r} \rangle$
$\tau_x \leftarrow \tau_2 \cdot x^2 + \tau_1 \cdot x + z^2\gamma; \mu \leftarrow \alpha + \rho \cdot x$          $\xrightarrow{\tau_x,\mu,\hat{t}}$
          $\xleftarrow{o}$ $o \leftarrow_{\$} \mathbb{Z}_p^*$
$\mathbf{h}' \leftarrow \mathbf{h}^{\mathbf{y}^{-n}}; u' \leftarrow u^o$          $\mathbf{h}' \leftarrow \mathbf{h}^{\mathbf{y}^{-n}}; u' \leftarrow u^o$
$P \leftarrow AS^x\mathbf{g}^{-z\cdot\mathbf{1}^n}\mathbf{h}'^{z\cdot\mathbf{y}^n + z^2\cdot\mathbf{2}^n}$          $P \leftarrow AS^x\mathbf{g}^{-z\cdot\mathbf{1}^n}\mathbf{h}'^{z\cdot\mathbf{y}^n + z^2\cdot\mathbf{2}^n}$
$P' \leftarrow h^{-\mu}P(u')^{\hat{t}}$          $P' \leftarrow h^{-\mu}P(u')^{\hat{t}}$
          $\mathsf{InPrd.P}((\mathbf{g}, \mathbf{h}', u', P'), (\mathbf{l}, \mathbf{r})) \Longleftrightarrow \mathsf{InPrd.V}(\mathbf{g}, \mathbf{h}', u', P') \to b$
          $R \leftarrow V^{z^2} g^{\delta(y,z)} T_1^x T_2^{x^2}$
          If $b = 1 \wedge g^{\hat{t}}h^{\tau_x} = R$ 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.

The prover and verifier for RngPf are defined in Figure 7. In this protocol the prover computes $\mathbf{a}_L$ – the n-bit representation of v, $\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n$ and commits to $\mathbf{a}_L, \mathbf{a}_R$ and proves to the verifier that $\mathbf{a}_L = \mathbf{a}_R - \mathbf{1}^n$, $\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{0}^n$ and $\langle \mathbf{a}_L, \mathbf{2}^n \rangle = v$. (The prover and the verifier of RngPf engage in InPrd in the final step to avoid the prover sending over vectors of length n). The first two equalities imply that $\mathbf{a}_L \in \{0, 1\}^n$. Combining this with the third equality gives that $v \in [0, 2^n - 1]$.

We shall prove the following theorem to establish an upper bound on the online srs-wee security for RngPf.

Theorem 4. *Let $\mathbb{G} = \{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}^+}$ be a family of groups of order $p = p(\lambda)$. Let* RngPf = RngPf[$\mathbb{G}$] *be the interactive argument as defined in Figure 7, for the relation R in (6). 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 $\lambda \in \mathbb{N}^+$*

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

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 (6). 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 $\lambda \in \mathbb{N}^+$*

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

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 3 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 CheckBad to define $\mathsf{BadCh}(\tau')$. The function CheckBad takes as input a partial extended transcript $[\tau']$ and a challenge $c \in \mathsf{Ch}$ and returns `true` if and only if $c \in \mathsf{BadCh}(\tau')$. For each verifier challenge in RngPf, there is a definition of CheckBad in Figure 8. Every 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 `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 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 `true`, for all the definitions of CheckBad, using the Schwartz-Zippel Lemma.

The function $\mathsf{CheckBad}(\tau', (y, z))$ returns `true` if any of $\mathsf{SZ}(f_i(Z), z)$ for $i = 1, 2, 3$ is `true` or if $\mathsf{SZ}(f_4(Y, Z), (y, z))$ is `true`. Since $f_1(Z)$ is a vector of n polynomials of degree 2, the fraction of z's in $\mathbb{Z}_p^*$ for which $\mathsf{SZ}(f_1(Z), z)$ is `true` is at most $2n/(p - 1)$ using the Schwartz-Zippel Lemma and the union bound. Similarly, the fraction of z's for which $\mathsf{SZ}(f_2(Z), z)$ and $\mathsf{SZ}(f_3(Z), z)$ is `true` is at most $2n/(p - 1)$ and $2/(p - 1)$ respectively. The polynomial $f_4(Y, Z)$ is a polynomial of degree at most $n + 1$. So, the fraction of (y, z)'s for which $\mathsf{SZ}(f_4(Y, Z), (y, z))$ is `true` is at most $(n + 1)/(p - 1)$. Using the union bound, the fraction of $y, z \in \mathbb{Z}_p^*$ for which $\mathsf{CheckBad}(\tau', (y, z))$ returns `true` is at most $(5n + 3)/(p - 1)$.

The function $\mathsf{CheckBad}(\tau', x)$ returns `true` if any of $\mathsf{SZ}(f_i(X), x)$ for $i = 1, 2, 3, 4$ is `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 `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 `true` is at most $2/(p-1)$. The fraction of x's for which $z^2 + t_{1V}x + t_{2V}x^2 = 0$ 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 `true` is at most $(4n+6)/(p-1)$.

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

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 $m \in \{1, \dots, \log n\}$.

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.

$$\begin{aligned}\tau =\big(&(n, \mathbf{g}, \mathbf{h}, u, g, h), V; (A, S), (y, z), (T_1, T_2), x, (\tau_x, \mu, \hat{t}), o, (L_1, R_1), x_1, (L_2, R_2), x_2, \dots, \\ &(L_{\log n}, R_{\log n}), x_{\log n}, (a, b)\big) \ .\end{aligned} \tag{7}$$

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

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

Observe from the definition of $\mathsf{RngPf.V}$ that if τ as defined in (7) is an accepting transcript,

$$V^{z^2} g^{\delta(y,z)} T_1^x T_2^{x^2} = g^{\hat{t}} h^{\tau_x} \ . \tag{8}$$

Now, e can plug in the representations of T_1, T_2 into (8) and compute $e_V, e_{\mathbf{g}}, e_{\mathbf{h}}, e_u, e_g, e_h$ 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

$$e_g = \hat{t} - \delta(y,z) - t_{1g}x - t_{2g}x^2 \ , \ e_V = z^2 + t_{1V}x + t_{2V}x^2 \ .$$

Note that the x for which $e_V = 0$ is in $\mathsf{BadCh}(\tau|_x)$. The procedure e (formally defined in Fig. 9) 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 $e_g/e_V \in [0, 2^n - 1]$. It follows from the description of e runs in time $O(n)$. Note that $\mathsf{RngPf.V}$ runs in time $O(n)$. Therefore, using Theorem 3, the time complexity $\mathcal{E}$ is $O(q \cdot n)$.

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 $\tau \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{RngPf}}$ if $\mathsf{h}([\tau])$ returns a trivial discrete logarithm relation, then $\mathsf{e}(\overline{[\tau]})$ returns a valid witness. Taking the contrapositive, we have that whenever $\mathsf{e}(\overline{[\tau]})$ fails to extract a valid witness for an accepting

```
Procedure CheckBad($[\tau'], (y,z)$):
//$[\tau'] = ((n, \mathbf{g}, \mathbf{h}, u, g, h), [V], ([A], [S]))$
$f_1(Z) \leftarrow v_{\mathbf{g}} Z^2$; $f_2(Z) \leftarrow v_{\mathbf{h}} Z^2$; $f_3(Z) \leftarrow v_u Z^2$; $f_4(Y,Z) \leftarrow Z^2(v_g - \langle a_{\mathbf{g}}, \mathbf{2}^n \rangle) - Z\langle a_{\mathbf{g}} - a_{\mathbf{h}} - \mathbf{1}^n, \mathbf{Y}^n \rangle - \langle a_{\mathbf{g}} \circ a_{\mathbf{h}}, \mathbf{Y}^n \rangle$
Return $\mathsf{SZ}(f_1(Z), z) \vee \mathsf{SZ}(f_2(Z), z) \vee \mathsf{SZ}(f_3(Z), z) \vee \mathsf{SZ}(f_4(Y,Z), (y,z))$

Procedure CheckBad($[\tau'], x$):
//$[\tau'] = ((n, \mathbf{g}, \mathbf{h}, u, g, h), [V], ([A], [S]), (y,z), ([T_1], [T_2]))$
If $z^2 + t_{1V}x + t_{2V}x^2 = 0$ then return true
$f_1(X) \leftarrow v_{\mathbf{g}}(z^2 + t_{1V}X + t_{2V}X^2) + t_{1\mathbf{g}}X + t_{2\mathbf{g}}X^2$; $f_2(X) \leftarrow v_{\mathbf{h}}(z^2 + t_{1V}X + t_{2V}X^2) + t_{1\mathbf{h}}X + t_{2\mathbf{h}}X^2$
$f_3(X) \leftarrow v_u(z^2 + t_{1V}X + t_{2V}X^2) + t_{1u}X + t_{2u}X^2$
$l(X) \leftarrow (a_{\mathbf{g}} - z \cdot \mathbf{1}^n) + s_{\mathbf{g}} \cdot X$; $r(X) \leftarrow \mathbf{y}^n \circ (a_{\mathbf{h}} + z \cdot \mathbf{1}^n + s_{\mathbf{h}} \cdot X) + z^2 \cdot \mathbf{2}^n$; $\delta(y,z) \leftarrow (z - z^2)\langle \mathbf{1}^n, \mathbf{y}^n \rangle - z^3 \langle \mathbf{1}^n, \mathbf{2}^n \rangle$
$f_4(X) \leftarrow v_g(z^2 + t_{1V}X + t_{2V}X^2) + \delta(y,z) + t_{1g}X + t_{2g}X^2 - \langle l(X), r(X) \rangle$
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 CheckBad($[\tau'], o$):
//$[\tau'] = ((n, \mathbf{g}, \mathbf{h}, u, g, h), [V], ([A], [S]), (y,z), ([T_1], [T_2]), x, (\tau_x, \mu, \hat{t}))$
$\mathbf{l} \leftarrow (a_{\mathbf{g}} - z \cdot \mathbf{1}^n) + s_{\mathbf{g}} \cdot x$; $\mathbf{r} \leftarrow (a_{\mathbf{h}} + x s_{\mathbf{h}} + z\mathbf{1}^n) \circ \mathbf{y}^n + z^2\mathbf{2}^n$; $f(O) \leftarrow O\hat{t} - O\langle \mathbf{l}, \mathbf{r} \rangle$
Return $\mathsf{SZ}(f(O), o)$

Procedure CheckBad($[\tau'], x_m$):
//$[\tau'] = ((n, \mathbf{g}, \mathbf{h}, u, g, h), [V], ([A], [S]), (y,z), ([T_1], [T_2]), x, (\tau_x, \mu, \hat{t}), o, ([L_1], [R_1]), x_1, \ldots, ([L_m], [R_m]))$
$p'_{\mathbf{g}} \leftarrow a_{\mathbf{g}} + x s_{\mathbf{g}} - z\mathbf{1}^n$; $p'_{\mathbf{h}} \leftarrow a_{\mathbf{h}} + x s_{\mathbf{h}} + \mathbf{y}^{-n} \circ (z\mathbf{y}^n + z^2\mathbf{2}^n)$; $p'_u \leftarrow a_u + x s_u + o\hat{t}$
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 ← false
For $t = 1, \ldots, m-1$ do for $j = 0, \ldots, n/2^t - 1$ do
  flag ← 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 ← 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 ← flag $\vee$ $\mathsf{SZ}\Big(f^u_m(X) - o \cdot \sum_{j=0}^{n/2^m - 1} f^{\mathbf{g}}_{m,j}(X) \cdot f^{\mathbf{h}}_{m,j}(X) \cdot y^j, x_m\Big)$
Return flag
```

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

```
Procedure e($\overline{[\tau]}$):
//$\overline{[\tau]} = ((n, \mathbf{g}, \mathbf{h}, u, g, h), V; ([A], [S]), (y,z), ([T_1], [T_2]), x, (\tau_x, \mu, \hat{t}), o, ([L_1], [R_1]), x_1, \ldots, ([L_{\log n}], [R_{\log n}]), x_{\log n},$
        $(a, b))$
If $z^2 + t_{1V}x + t_{2V}x^2 = 0$ then return $\perp$
$\delta(y,z) \leftarrow (z - z^2)\langle \mathbf{1}^n, \mathbf{y}^n \rangle - z^3 \langle \mathbf{1}^n, \mathbf{2}^n \rangle$
$v^* \leftarrow \frac{\hat{t} - \delta(y,z) - t_{1g}x - t_{2g}x^2}{z^2 + t_{1V}x + t_{2V}x^2}$; $\gamma^* \leftarrow \frac{\tau_x - t_{1h}x - t_{2h}x^2}{z^2 + t_{1V}x + t_{2V}x^2}$; Return $(v^*, \gamma^*)$
```

Fig. 9. The function e for RngPf.

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 (7), be an accepting transcript. The following equality must hold since τ is an accepting transcript.

$$V^{z^2} g^{\delta(y,z)} T_1^x T_2^{x^2} = g^{\hat{t}} h^{\tau_x} \ .$$

Like e, h 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 $V^{e_V} = \mathbf{g}^{e_{\mathbf{g}}} \mathbf{h}^{e_{\mathbf{h}}} u^{e_u} g^{e_g} h^{e_h}$. Additionally, h has the representation of V- plugging that in h can 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 $e^{(1)}_{\mathbf{g}}, e^{(1)}_{\mathbf{h}}, e^{(1)}_g, e^{(1)}_h, e^{(1)}_u$. Note that $e^{(1)}_{\mathbf{g}} = e_{\mathbf{g}} - e_V v_{\mathbf{g}}$, $e^{(1)}_{\mathbf{h}} = e_{\mathbf{h}} - e_V v_{\mathbf{h}}$, $e^{(1)}_g = e_g - e_V v_g$, $e^{(1)}_h = e_h - e_V v_h$ and $e^{(1)}_u = e_u - e_V v_u$.

Again since τ is an accepting transcript, $\mathsf{InPrd.V}$ must have returned 1 and hence the following equality must hold.

$$P^{(\log n)} = (\mathbf{g}^{(\log n)})^a (\mathbf{h}^{(\log n)})^b u^{ab} \ . \tag{9}$$

Using the definition of $P^{(i)}$'s in $\mathsf{InPrd.V}$, the left hand side of (9) can be written as

$$\left(\prod_{i=1}^{\log n} L_i^{x_i^2}\right) h^{-\mu} A S^x \mathbf{g}^{-z\cdot\mathbf{1}^n} ((\mathbf{h})^{\mathbf{y}^{-n}})^{z\cdot\mathbf{y}^n + z^2\cdot\mathbf{2}^n} (u^o)^{\hat{t}} \left(\prod_{i=1}^{\log n} R_i^{x_i^{-2}}\right) \ .$$

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. Using the definition of $\mathbf{g}^{(i)}$'s and $\mathbf{h}^{(i)}$'s in $\mathsf{InPrd.V}$, the right hand side of (9) can be written as

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

Plugging these into (9), 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 $O(q \cdot n)$.

```
Procedure h([τ]):
$//[\tau] = \big((n, \mathbf{g}, \mathbf{h}, u, g, h), [V]; ([A], [S]), (y, z), ([T_1], [T_2]), x, (\tau_x, \mu, \hat{t}), o, ([L_1], [R_1]), x_1, \ldots,$
        $([L_{\log n}], [R_{\log n}]), x_{\log n}, (a, b)\big)$
$\delta(y, z) \leftarrow (z - z^2)\langle \mathbf{1}^n, \mathbf{y}^n \rangle - z^3 \langle \mathbf{1}^n, \mathbf{2}^n \rangle$
$e_{\mathbf{g}}^{(1)} \leftarrow v_{\mathbf{g}}(z^2 + t_{1V}x + t_{2V}x^2) + t_{1\mathbf{g}}x + t_{2\mathbf{g}}x^2$; $e_{\mathbf{h}}^{(1)} \leftarrow v_{\mathbf{h}}(z^2 + t_{1V}x + t_{2V}x^2) + t_{1\mathbf{h}}x + t_{2\mathbf{h}}x^2$
$e_u^{(1)} \leftarrow v_u(z^2 + t_{1V}x + t_{2V}x^2) + t_{1u}x + t_{2u}x^2$; $e_g^{(1)} \leftarrow v_g(z^2 + t_{1V}x + t_{2V}x^2) + \delta(y, z) + t_{1g}x + t_{2g}x^2 - \hat{t}$
$e_h^{(1)} \leftarrow v_h(z^2 + t_{1V}x + t_{2V}x^2) + t_{1h}x + t_{2h}x^2 - \tau_x$
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 $(e_{\mathbf{g}}^{(1)}, e_{\mathbf{h}}^{(1)}, e_u^{(1)}, e_g^{(1)}, e_h^{(1)})$
$p'_V \leftarrow a_V + xs_V$; $p'_{\mathbf{g}} \leftarrow (a_{\mathbf{g}} + a_V v_{\mathbf{g}}) + x(s_{\mathbf{g}} + s_V v_{\mathbf{g}}) - z\mathbf{1}^n$
$p'_{\mathbf{h}} \leftarrow (a_{\mathbf{h}} + a_V v_{\mathbf{h}}) + x(s_{\mathbf{h}} + s_V v_{\mathbf{h}}) + \mathbf{y}^{-n} \circ (z\mathbf{y}^n + z^2\mathbf{2}^n)$; $p'_g \leftarrow (a_g + a_V v_g) + x(s_g + s_V v_g)$
$p'_h \leftarrow (a_h + a_V v_h) + x(s_h + s_V v_h) - \mu$; $p'_u \leftarrow (a_u + a_V v_u) + x(s_u + s_V v_u) + o\hat{t}$
For $k = 0$ to $n - 1$ do
    $e_{g_{k+1}}^{(2)} \leftarrow \left(p'_{g_{1+k}} + p'_V v_{g_{1+k}} + \sum_{i=1}^{\log n} (l_{ig_{1+k}} + l_{iV} v_{g_{1+k}})x_i^2 + (r_{ig_{1+k}} + r_{iV} v_{g_{1+k}})x_i^{-2}\right) - a \cdot \left(\prod_{i=1}^{\log n} x_i^{(-1)^{1-\mathsf{bit}(k,i,\log n)}}\right)$
    $e_{h_{k+1}}^{(2)} \leftarrow \left(p'_{h_{1+k}} + p'_V v_{h_{1+k}} + \sum_{i=1}^{\log n} (l_{ih_{1+k}} + l_{iV} v_{h_{1+k}})x_i^2 + (r_{ih_{1+k}} + r_{iV} v_{h_{1+k}})x_i^{-2}\right) - by^{(-(k))} \cdot \left(\prod_{i=1}^{\log n} x_i^{(-1)^{\mathsf{bit}(k,i,\log n)}}\right)$
$e_{\mathbf{g}}^{(2)} \leftarrow (e_{g_1}^{(2)}, \ldots, e_{g_n}^{(2)})$; $e_{\mathbf{h}}^{(2)} \leftarrow (e_{h_1}^{(2)}, \ldots, e_{h_n}^{(2)})$
$e_u^{(2)} \leftarrow \left(p'_u + p'_V v_u + \sum_{i=1}^{\log n} (l_{iu} + l_{iV} v_u)x_i^2 + (r_{iu} + r_{iV} v_u)x_i^{-2}\right) - o \cdot ab$
$e_g^{(2)} \leftarrow \left(\sum_{i=1}^{\log n} (l_{ig} + l_{iV} v_g)x_i^2 + (r_{ig} + r_{iV} v_g)x_i^{-2}\right) + p'_g + p'_V v_g$
$e_h^{(2)} \leftarrow \left(\sum_{i=1}^{\log n} (l_{ih} + l_{iV} v_h)x_i^2 + (r_{ih} + r_{iV} v_h)x_i^{-2}\right) + p'_h + p'_V v_h$
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.

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 $\tau \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{RngPf}}$ if $\mathsf{h}([\tau])$ returns a trivial discrete logarithm relation, then $\mathsf{e}(\overline{[\tau]})$ returns a valid witness.

Lemma 6. *Let τ, as defined in (7), be an accepting transcript of* RngPf *such that $\tau \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{RngPf}}$. If $\mathsf{h}([\tau])$ returns $(\mathbf{0}^n, \mathbf{0}^n, 0, 0, 0)$ then $\mathsf{e}(\overline{[\tau]})$ returns (v^*, γ^*) such that*

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

Proving this lemma would conclude the proof of Theorem 4.

□

Proof (Lemma 6). The proof of this lemma will proceed by deriving several equalities given that $\mathsf{h}(\tau)$ returns a trivial discrete logarithm and that $\tau \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{RngPf}}$.

In order to prove $g^{v^*} h^{\gamma^*} = V$ and $v^* \in [0, 2^n - 1]$, it suffices to show that $e_{\mathbf{g}} = e_{\mathbf{h}} = \mathbf{0}^n$, $e_u = 0$ and $e_g/e_V \in [0, 2^n - 1]$ as argued before. Recall that, $e_{\mathbf{g}}^{(1)} = e_{\mathbf{g}} - e_V v_{\mathbf{g}}$, $e_{\mathbf{h}}^{(1)} = e_{\mathbf{h}} - e_V v_{\mathbf{h}}$, $e_g^{(1)} = e_g - e_V v_g$, $e_h^{(1)} = e_h - e_V v_h$ and $e_u^{(1)} = e_u - e_V v_u$.

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 $\tau \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{RngPf}}$ we have that $x \notin \mathsf{BadCh}(\tau|_x)$. So, $z^2 + t_{1V}x + t_{2V}x^2 \neq 0$, i.e., $e_V \neq 0$. Since $\mathsf{h}([\tau])$ returns $(\mathbf{0}^n, \mathbf{0}^n, 0, 0, 0)$, we have that $(e_{\mathbf{g}}^{(1)}, e_{\mathbf{h}}^{(1)}, e_g^{(1)}, e_h^{(1)}, e_u^{(1)}) = (e_{\mathbf{g}}^{(2)}, e_{\mathbf{h}}^{(2)}, e_g^{(2)}, e_h^{(2)}, e_u^{(2)}) = (\mathbf{0}^n, \mathbf{0}^n, 0, 0, 0)$.

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

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

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 $v_{\mathbf{g}}z^2 = \mathbf{0}^n$. Again since $\tau \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{RngPf}}$, we have that $(y, z) \notin \mathsf{BadCh}(\tau|_{(y,z)})$. Therefore, $\mathsf{SZ}(f_1(Z), z)$ is `false` where f_1 is as defined in $\mathsf{CheckBad}(\tau', (y, z))$. Since we have here that $f_1(z) = 0$, the polynomial $f_1(Z)$ is the zero vector polynomial. In particular its constant term $v_{\mathbf{g}} = \mathbf{0}^n$. Using $e_{\mathbf{g}}^{(1)} = e_{\mathbf{g}} - e_V v_{\mathbf{g}}$, we can conclude that $e_{\mathbf{g}} = \mathbf{0}^n$.

Similarly using $e_{\mathbf{h}}^{(1)} = \mathbf{0}^n$ we can show that $v_{\mathbf{h}} = \mathbf{0}^n$ and $e_{\mathbf{h}} = \mathbf{0}^n$. Using $e_u^{(1)} = 0$ we can show that that $v_u = 0$ and $e_u = v_u e_V = 0$.

Writing out the expression for $e_g^{(1)}$ we have $v_g(z^2 + t_{1V}x + t_{2V}x^2) + \delta(y, z) + t_{1g}x + t_{2g}x^2 - \hat{t} = 0$. Hence,

$$\hat{t} = v_g(z^2 + t_{1V}x + t_{2V}x^2) + \delta(y, z) + t_{1g}x + t_{2g}x^2 \ . \tag{10}$$

Further, using $e_g^{(1)} = e_g - e_V v_g$, we get that $v_g = e_g / e_V$.

So we have shown that $e_{\mathbf{g}} = \mathbf{0}^n$, $e_{\mathbf{h}} = \mathbf{0}^n$, $e_g/e_V = v_g$ and $e_u = 0$. Now, we need to show $e_g/e_V \in [0, 2^n - 1]$.

Since $v_{\mathbf{g}} = \mathbf{0}^n, v_{\mathbf{h}} = \mathbf{0}^n, v_u = 0$ the $p'_{\mathbf{g}}, p'_{\mathbf{h}}, p'_u$ as defined in h can be simplified to $p'_{\mathbf{g}} = a_{\mathbf{g}} + xs_{\mathbf{g}} - z\mathbf{1}^n, p'_{\mathbf{h}} = a_{\mathbf{h}} + xs_{\mathbf{h}} + \mathbf{y}^{-n} \circ (z\mathbf{y}^n + z^2\mathbf{2}^n), p'_u = a_u + xs_u + o\hat{t}$.

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

$$\left(\sum_{i=1}^{\log n} (l_{ig_{1+k}} x_i^2 + r_{ig_{1+k}} x_i^{-2}) + p'_{g_{1+k}} \right) - a \cdot \left(\prod_{i=1}^{\log n} x_i^{(-1)^{1-\mathsf{bit}(k,i,\log n)}} \right) = 0 \ . \tag{11}$$

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

$$\left(\sum_{i=1}^{\log n} (l_{ih_{1+k}} x_i^2 + r_{ih_{1+k}} x_i^{-2}) + p'_{h_{1+k}} \right) - by^{(-(k))} \cdot \left(\prod_{i=1}^{\log n} x_i^{(-1)^{\mathsf{bit}(k,i,\log n)}} \right) = 0 \ . \tag{12}$$

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

$$\left(\sum_{i=1}^{\log n} (l_{iu} x_i^2 + r_{iu} x_i^{-2}) \right) + p'_u - o \cdot ab = 0 \ . \tag{13}$$

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

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

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

```
Procedure Bad(params, $x, m$):
//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
    flag $\leftarrow$ flag $\vee$ SZ$(f^{\mathbf{g}}_{m,j}(X) \cdot x_t^2 - f^{\mathbf{g}}_{m,j+n/2^t}(X), x) \vee$ 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
  flag $\leftarrow$ flag $\vee$ SZ$(f^{\mathbf{g}}_{m,j}(X) \cdot X^2 - f^{\mathbf{g}}_{m,j+n/2^m}(X), x) \vee$ SZ$(f^{\mathbf{h}}_{m,j}(X) - f^{\mathbf{h}}_{m,j+n/2^m}(X) \cdot X^2, x)$
flag $\leftarrow$ flag $\vee$ SZ $\left(f^u_m(X) - o \cdot \sum_{j=0}^{n/2^m-1} f^{\mathbf{g}}_{m,j}(X) \cdot f^{\mathbf{h}}_{m,j}(X) \cdot y^j, x\right)$
Return flag
```

Fig. 11. The function Bad for Lemma 8.

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

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 (Lemma 7). 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* 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* Bad(params, x_i, i) $=$ `false` *for $i = 1, \ldots, \log n$ where* Bad *is defined in Figure 11. Suppose, the following equalities hold.*

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

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

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

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

3.
$$\left(\sum_{i=1}^{\log n} (l_{iu} x_i^2 + r_{iu} x_i^{-2})\right) + p'_u - o \cdot ab = 0 \ .$$

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

Let 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 $\mathsf{Bad}(\text{params}, x, j)$ returns `true` if and only if $x \in \mathsf{BadCh}(\tau|_{x_j})$. Therefore, we have that $x_1, \dots, x_{\log n}$ 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
$$o \cdot \langle p'_{\mathbf{g}}, p'_{\mathbf{h}} \circ \mathbf{y}^n \rangle = p'_u \ .$$
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 (11) to (13) hold, using Lemma 7, we get $o\langle p'_{\mathbf{g}}, p'_{\mathbf{h}} \circ \mathbf{y}^n \rangle = p'_u$. Plugging in the values of $p'_{\mathbf{g}}, p'_{\mathbf{h}}, p'_u$ (using the fact $v_{\mathbf{g}} = v_{\mathbf{h}} = \mathbf{0}^n, v_u = 0$ and simplifying) we get

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

Since $\tau \notin \mathcal{T}^{\mathsf{RngPf}}_{\mathsf{BadCh}}$, we have that $o \notin \mathsf{BadCh}(\tau|_o)$. Therefore, $\mathsf{SZ}(f(O), o)$ is `false` where f is as defined in $\mathsf{CheckBad}(\tau', o)$. Since we have here that $f(o) = 0$, the polynomial $f(O)$ must be the zero polynomial. In particular its O 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 (10), we have that

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

Since $\tau \notin \mathcal{T}^{\mathsf{RngPf}}_{\mathsf{BadCh}}$, 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 constant term must be zero, i.e.,

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

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

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

Since $\tau \notin \mathcal{T}^{\mathsf{RngPf}}_{\mathsf{BadCh}}$, we have that $(y,z) \notin \mathsf{BadCh}(\tau|_{(y,z)})$. Therefore, $\mathsf{SZ}(f_4(Y,Z),(y,z))$ is `false` where f_4 is as defined in $\mathsf{CheckBad}(\tau', (y,z))$. Since we have here that $f_4(y,z) = 0$, the polynomial

$f_4(Y, Z)$ is the zero polynomial. Therefore, equating all the coefficients of $f_4(Y, Z)$ to zero, 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, $e_V, e_g, e_{\mathbf{g}}, e_{\mathbf{h}}$ output by $\mathsf{e}(\lceil \tau \rceil)$ satisfy $e_g/e_V \in [0, 2^n - 1]$, $e_{\mathbf{g}} = e_{\mathbf{h}} = \mathbf{0}^n$ and $e_u = 0$. So, $V = g^{v^*} h^{\gamma^*}$ and $v^* \in [0, 2^n - 1]$.

This concludes the proof of Lemma 6.

□

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

$$R' = \left\{ (n \in \mathbb{N}, g, V \in \mathbb{G}, v \in \mathbb{Z}_p) : g^v = V \wedge v \in [0, 2^n - 1] \right\} , \tag{14}$$

by fixing γ to 0.

We shall construct a cheating prover $\mathcal{P}$ (that makes $O(q)$ queries to $\mathbf{O}_{\mathrm{ext}}$) 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 (14) 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 $\mathbf{O}_{\mathrm{ext}}$ 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 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 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, $\tau_x = \tau_1 x + \tau_2 x^2$, $R = V^{z^2} g^{\delta(y,z)} T_1^x T_2^{x^2} = g^{z^2 v + t_1 x + t_2 x^2 + \delta(y,z)} h^{\tau_1 x + \tau_2 x^2}$. Now since $\hat{t} = t_0 + t_1 x + t_2 x^2$ we have

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

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

$$R = g^{z^2 v + t_1 x + t_2 x^2 + \delta(y,z)} h^{\tau_1 x + \tau_2 x^2} = g^{\hat{t}} h^{\tau_x} \,.$$

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 ACSPf $=$ 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

$$R = \Big\{ \big((n, Q \in \mathbb{N}), (\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O \in \mathbb{Z}_p^{Q \times n}, \mathbf{c} \in \mathbb{Z}_p^Q), (\mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O \in \mathbb{Z}_p^n)\big) : \\ \mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O \wedge \mathbf{W}_L \cdot \mathbf{a}_L + \mathbf{W}_R \cdot \mathbf{a}_R + \mathbf{W}_O \cdot \mathbf{a}_O = \mathbf{c} \Big\} \,. \tag{15}$$

$\mathsf{ACSPf.P}(((n, Q, \mathbf{g}, \mathbf{h}, g, h, u),$ $(\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c})), (\mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O))$		$\mathsf{ACSPf.V}((n, Q, \mathbf{g}, \mathbf{h}, g, h, u),$ $(\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c}))$
$\alpha \leftarrow_{\$} \mathbb{Z}_p; A \leftarrow h^{\alpha}\mathbf{g}^{\mathbf{a}_L}\mathbf{h}^{\mathbf{a}_R}$		
$\mathbf{s}_L \leftarrow_{\$} \mathbb{Z}_p^n; \mathbf{s}_R \leftarrow_{\$} \mathbb{Z}_p^n$		
$\alpha, \beta, \rho \leftarrow_{\$} \mathbb{Z}_p; S \leftarrow h^{\rho}\mathbf{g}^{\mathbf{s}_L}\mathbf{h}^{\mathbf{s}_R}$		
$A_I \leftarrow h^{\alpha}\mathbf{g}^{\mathbf{a}_L}\mathbf{h}^{\mathbf{a}_R}; A_O \leftarrow h^{\beta}\mathbf{g}^{\mathbf{a}_O}$	$\xrightarrow{A_I, A_O, S}$	
	$\xleftarrow{y, z}$	$y, z \leftarrow_{\$} \mathbb{Z}_p^*$
$l(X) \leftarrow \mathbf{a}_L X + \mathbf{a}_O X^2 + \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_R) \cdot X$ $+ \mathbf{s}_L X^3$		$\delta(y, z) \leftarrow \langle \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_R), \mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_L \rangle$
$r(X) \leftarrow \mathbf{y}^n \circ \mathbf{a}_R \cdot X - \mathbf{y}^n + \mathbf{z}_{[1:]}^{Q+1} \cdot (\mathbf{W}_L \cdot X + \mathbf{W}_O)$ $+ \mathbf{y}^n \circ \mathbf{s}_R \cdot X^3$		
$t(X) \leftarrow \langle l(X), r(X) \rangle = \sum_{i=1}^{6} t_i X^i$		
$\tau_i \leftarrow_{\$} \mathbb{Z}_p$ for $i \in \{1, 3, 4, 5, 6\}$		
$T_i \leftarrow g^{t_i} h^{\tau_i}$ for $i \in \{1, 3, 4, 5, 6\}$	$\xrightarrow{T_1, T_3, T_4, T_5, T_6}$	
	$\xleftarrow{x}$	$x \leftarrow_{\$} \mathbb{Z}_p^*$
$\mathbf{l} \leftarrow l(x); \mathbf{r} \leftarrow r(x); \hat{t} \leftarrow \langle \mathbf{l}, \mathbf{r} \rangle$		
$\tau_x \leftarrow \tau_1 \cdot x + \sum_{i=3}^{6} \tau_i \cdot x^i$		
$\mu \leftarrow \alpha \cdot x + \beta \cdot x^2 + \rho \cdot x^3$	$\xrightarrow{\tau_x, \mu, \hat{t}}$	
	$\xleftarrow{o}$	$o \leftarrow_{\$} \mathbb{Z}_p^*$
$\mathbf{h}' \leftarrow \mathbf{h}^{\mathbf{y}^{-n}}; u' \leftarrow u^o$		$\mathbf{h}' \leftarrow \mathbf{h}^{\mathbf{y}^{-n}}; u' \leftarrow u^o$
$W_L \leftarrow \mathbf{h}'^{\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_L}$		$W_L \leftarrow \mathbf{h}'^{\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_L}$
$W_R \leftarrow \mathbf{g}^{\mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_R)}$		$W_R \leftarrow \mathbf{g}^{\mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_R)}$
$W_O \leftarrow \mathbf{h}'^{\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_O}$		$W_O \leftarrow \mathbf{h}'^{\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_O}$
$P \leftarrow A_I^x \cdot A_O^{x^2} \cdot \mathbf{h}'^{-\mathbf{y}^n} \cdot W_L^x \cdot W_R^x \cdot W_O \cdot S^{x^3}$		$P \leftarrow A_I^x \cdot A_O^{x^2} \cdot \mathbf{h}'^{-\mathbf{y}^n} \cdot W_L^x \cdot W_R^x \cdot W_O \cdot S^{x^3}$
$P' \leftarrow h^{-\mu} P (u')^{\hat{t}}$		$P' \leftarrow h^{-\mu} P (u')^{\hat{t}}$
$\mathsf{InPrd.P}((\mathbf{g}, \mathbf{h}', u', P'), (\mathbf{l}, \mathbf{r}))$	$\Longleftrightarrow$	$\mathsf{InPrd.V}(\mathbf{g}, \mathbf{h}', u', P') \rightarrow b$
		$R \leftarrow g^{x^2(\delta(y,z) + \langle \mathbf{z}_{[1:]}^{Q+1}, \mathbf{c} \rangle)} \cdot T_1^x \cdot \prod_{i=3}^{6} T_i^{x^i}$
		If $b = 1 \wedge g^{\hat{t}} h^{\tau_x} = R$ then
		Return 1
		Return 0

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

We note that in [BBB+18], an argument for a more generalized relation was given of which this is a special case. The generalized relation contained additional commitments. We are able to prove srs-wee only for the above relation. Having many different commitments with generators g, h does not allow us to build an online extractor. Our proof would work if the different commitments were made with different generators. 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 on input 1^λ returns positive integers n, Q and 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 (15). 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* ACSPf = ACSPf[$\mathbb{G}$] *be the interactive argument as defined in Figure 12, for the relation R in (15). 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 the following corollary.

Corollary 2. *Let $\mathbb{G} = \{\mathbb{G}_\lambda\}_{\lambda\in\mathbb{N}^+}$ be a family of groups of order $p = p(\lambda)$. Let* ACSPf = ACSPf[$\mathbb{G}$] *be the interactive argument as defined in Figure 12, for the relation R in (15). Let* $\mathsf{FS}^{\mathbf{RO}}$[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 $\lambda \in \mathbb{N}^+$*

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

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 next prove Theorem 6 – the proof is similar to the proof of Theorem 4.

Proof (Theorem 6). In order to prove this theorem, we invoke Theorem 3 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 BadCh(τ') 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 [τ'] and a challenge $c \in$ Ch and returns `true` if and only if $c \in$ BadCh(τ'). For each verifier challenge in ACSPf, there is a definition of CheckBad in Figure 8. Every 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 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 |BadCh(τ')|/|Ch|. To this end, we compute an upper bound on the maximum fraction of c's in Ch for which CheckBad(τ', c) will return `true`, for all the definitions of CheckBad, using the Schwartz-Zippel Lemma.

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

**Procedure** CheckBad($[\tau'], x$):
$//[\tau'] = ((n, Q, \mathbf{g}, \mathbf{h}, u, g, h), (\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c}); ([A_I], [A_O], [S]), (y, z), ([T_1], [T_3], [T_4], [T_5], [T_6]))$
$f_1(X) \leftarrow t_{1\mathbf{g}}X + \sum_{i=3}^{6} t_{i\mathbf{g}}X^i$; $f_2(X) \leftarrow t_{1\mathbf{h}}X + \sum_{i=3}^{6} t_{i\mathbf{h}}X^i$; $f_3(X) \leftarrow t_{1u}X + \sum_{i=3}^{6} t_{iu}X^i$
$l(X) \leftarrow a_{I\mathbf{g}} \cdot X + a_{O\mathbf{g}} \cdot X^2 + \mathbf{y}^{-n} \circ (\mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_R) \cdot X + s_{\mathbf{h}} \cdot X^3$; $r(X) \leftarrow \mathbf{y}^n \circ a_{I\mathbf{h}} \cdot X - \mathbf{y}^n + \mathbf{z}^{Q+1}_{[1:]} \cdot (\mathbf{W}_L \cdot X + \mathbf{W}_O) + \mathbf{y}^n \circ s_{\mathbf{h}}$
$\delta(y, z) \leftarrow \langle \mathbf{y}^{-n} \circ (\mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_R), \mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_L \rangle$; $f_4(X) \leftarrow X^2(\delta(y, z) + \langle \mathbf{z}^{Q+1}_{[1:]}, \mathbf{c} \rangle) + t_{1g}X + \sum_{i=1}^{3} t_{ig}X^i - \langle l(X), r(X) \rangle$
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** CheckBad($[\tau'], o$):
$//[\tau'] = ((n, Q, \mathbf{g}, \mathbf{h}, u, g, h), (\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c}); ([A_I], [A_O], [S]), (y, z), ([T_1], [T_3], [T_4], [T_5], [T_6]), x, (\tau_x, \mu, \hat{t}))$
$\mathbf{l} \leftarrow a_{I\mathbf{g}} \cdot x + a_{O\mathbf{g}} \cdot x^2 + \mathbf{y}^{-n} \circ (\mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_R) \cdot x + s_{\mathbf{h}} \cdot x^3$
$\mathbf{r} \leftarrow \mathbf{y}^n \circ a_{I\mathbf{h}} \cdot x - \mathbf{y}^n + \mathbf{z}^{Q+1}_{[1:]} \cdot (\mathbf{W}_L \cdot x + \mathbf{W}_O) + \mathbf{y}^n \circ s_{\mathbf{h}}$; $f(O) \leftarrow O\hat{t} - O\langle \mathbf{l}, \mathbf{r} \rangle$
Return $\mathsf{SZ}(f(O), o)$

**Procedure** CheckBad($[\tau'], x_m$):
$//[\tau'] = ((n, Q, \mathbf{g}, \mathbf{h}, u, g, h), (\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c}); ([A_I], [A_O], [S]), (y, z), ([T_1], [T_3], [T_4], [T_5], [T_6]), x, (\tau_x, \mu, \hat{t}), o,$
          $([L_1], [R_1]), x_1, \ldots, ([L_m], [R_m]))$
$p'_{\mathbf{g}} \leftarrow a_{I\mathbf{g}} \cdot x + a_{O\mathbf{g}} \cdot x^2 + \mathbf{y}^{-n} \circ (\mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_R) \cdot x + s_{\mathbf{g}} \cdot x^3$; $p'_u \leftarrow a_{Iu} \cdot x + a_{Iu} \cdot x^2 + s_u \cdot x^3 + o\hat{t}$
$p'_{\mathbf{h}} \leftarrow a_{I\mathbf{h}} \cdot x + a_{O\mathbf{h}} \cdot x^2 - \mathbf{1}^n + \mathbf{y}^{-n} \circ (\mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_L) \cdot x + \mathbf{y}^{-n} \circ (\mathbf{z}^{Q+1}_{[1:]} \cdot \mathbf{W}_O) + s_{\mathbf{g}} \cdot x^3$
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 ← false
For $t = 1, \ldots, m-1$ do
  For $j = 0, \ldots, n/2^t - 1$ do
    flag ← 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 ← 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 ← flag $\vee$ $\mathsf{SZ}\left(f^u_m(X) - o \cdot \sum_{j=0}^{n/2^m-1} f^{\mathbf{g}}_{m,j}(X) \cdot f^{\mathbf{h}}_{m,j}(X) \cdot y^j, x_m\right)$
Return flag
```

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

```
Procedure $\mathsf{e}([\overline{\tau}])$:
$//[\overline{\tau}] = ((n, Q, \mathbf{g}, \mathbf{h}, u, g, h), (\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c}); ([A], [S]), (y, z), ([T_1], [T_3], [T_4], [T_5], [T_6]), x, (\tau_x, \mu, \hat{t}), o,$
        $([L_1], [R_1]), x_1, \ldots, ([L_{\log n}], [R_{\log n}]), x_{\log n}, (a, b))$
Return $(a_{I\mathbf{g}}, a_{I\mathbf{h}}, a_{O\mathbf{g}})$
```

Fig. 14. The function e for ACSPf.

The function $\mathsf{CheckBad}(\tau', (y, z))$ returns `true` if $\mathsf{SZ}(f(Y, Z), (y, z))$ is `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 `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 `true` is at most $(n + 1)/(p - 1)$.

The function $\mathsf{CheckBad}(\tau', x)$ returns `true` if at least one of $\mathsf{SZ}(f_i(X), x)$ is `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 `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 `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 `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 `true` is at most $(12n + 10)/((p - 1))$.

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

The function $\mathsf{CheckBad}(\tau', x_m)$ for $m \in \{1, \ldots, \log n\}$ 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, $2n/2^m$ 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 `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 `true` for any partial transcript τ' is upper bounded by $(14n + 8)/(p - 1)$, i.e., in the context of Theorem 3, $\varepsilon \leqslant \frac{14n+8}{p-1}$.

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 $O(n)$. Note that ACSPf.V runs in time $O(n)$. Therefore, using Theorem 3, the time complexity of $\mathcal{E}$ is $O(q \cdot n)$.

In order to complete our proof we need compute an upper bound on $p_{\mathsf{fail}}(\mathsf{ACSPf}, \mathcal{P}_{\mathsf{alg}}, \mathsf{e}, R, \lambda)$. 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.

```
Procedure h([τ]):
$//[\tau] = \big((n, Q, \mathbf{g}, \mathbf{h}, u, g, h), (\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c}); ([A], [S]), (y, z), ([T_1], [T_3], [T_4], [T_5], [T_6]), x, (\tau_x, \mu, \hat{t}), o,$
        $([L_1], [R_1]), x_1, \dots, ([L_{\log n}], [R_{\log n}]), x_{\log n}, (a, b)\big)$
$\delta(y,z) \leftarrow \langle \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_R), \mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_L \rangle; e_{\mathbf{g}}^{(1)} \leftarrow t_{1\mathbf{g}}x + \sum_{i=3}^{6} t_{i\mathbf{g}}x^i; e_{\mathbf{h}}^{(1)} \leftarrow t_{1\mathbf{h}}x + \sum_{i=3}^{6} t_{i\mathbf{h}}x^i; e_u^{(1)} \leftarrow t_{1u}x + \sum_{i=3}^{6} t_{iu}x^i$
$e_g^{(1)} \leftarrow x^2(\delta(y,z) + \langle \mathbf{z}_{[1:]}^{Q+1}, \mathbf{c} \rangle) + t_{1g}x + \sum_{i=3}^{6} t_{ig}x^i - \hat{t}; e_h^{(1)} \leftarrow t_{1h}x + \sum_{i=3}^{6} t_{ih}x^i - \tau_x$
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 $(e_{\mathbf{g}}^{(1)}, e_{\mathbf{h}}^{(1)}, e_u^{(1)}, e_g^{(1)}, e_h^{(1)})$
$p'_{\mathbf{g}} \leftarrow a_{I\mathbf{g}} \cdot x + a_{O\mathbf{g}} \cdot x^2 + \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_R) \cdot x + s_{\mathbf{g}} \cdot x^3$
$p'_{\mathbf{h}} \leftarrow a_{I\mathbf{h}} \cdot x + a_{O\mathbf{h}} \cdot x^2 - \mathbf{1}^n + \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_L) \cdot x + \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_O) + s_{\mathbf{g}} \cdot x^3$
$p'_g \leftarrow a_{Ig} \cdot x + a_{Ig} \cdot x^2 + s_g \cdot x^3; p'_h \leftarrow a_{Ih} \cdot x + a_{Ih} \cdot x^2 + s_h \cdot x^3 - \mu; p'_u \leftarrow a_{Iu} \cdot x + a_{Iu} \cdot x^2 + s_u \cdot x^3 + o\hat{t}; p'_V \leftarrow a_V + xs_V$
For $k = 0$ to $n-1$ do
  $e_{g_{k+1}}^{(2)} \leftarrow \left( \sum_{i=1}^{\log n} (l_{ig_{1+k}} x_i^2 + r_{ig_{1+k}} x_i^{-2}) + p'_{g_{1+k}} \right) - a \cdot \left( \prod_{i=1}^{\log n} x_i^{(-1)^{1-\mathsf{bit}(k,i,\log n)}} \right)$
  $e_{h_{k+1}}^{(2)} \leftarrow \left( \sum_{i=1}^{\log n} (l_{ih_{1+k}} x_i^2 + r_{ih_{1+k}} x_i^{-2}) + p'_{h_{1+k}} \right) - by^{(-(k))} \cdot \left( \prod_{i=1}^{\log n} x_i^{(-1)^{\mathsf{bit}(k,i,r)}} \right)$
$e_{\mathbf{g}}^{(2)} \leftarrow (e_{g_1}^{(2)}, \dots, e_{g_n}^{(2)}); e_{\mathbf{h}}^{(2)} \leftarrow (e_{h_1}^{(2)}, \dots, e_{h_n}^{(2)})$
$e_u^{(2)} \leftarrow \left( \sum_{i=1}^{\log n} (l_{iu} x_i^2 + r_{iu} x_i^{-2}) + p'_u \right) - o \cdot ab; e_g^{(2)} \leftarrow \left( \sum_{i=1}^{\log n} l_{ig} x_i^2 + r_{ig} x_i^{-2} \right) + p'_g; e_h^{(2)} \leftarrow \left( \sum_{i=1}^{\log n} l_{ih} x_i^2 + r_{ih} x_i^{-2} \right) + p'_h$
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.

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.

$$\tau = \big((n, Q, \mathbf{g}, \mathbf{h}, u, g, h), (\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c}); (A_I, A_O, S), (y, z), (T_1, T_3, T_4, T_5, T_6), x, (\tau_x, \mu, \hat{t}), o,$$
$$(L_1, R_1), x_1, (L_2, R_2), x_2, \dots, (L_{\log n}, R_{\log n}), x_{\log n}, (a, b)\big) \ . \tag{16}$$

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

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

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

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

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 – if on an accepting transcript τ such that $\tau \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{ACSPf}}$ if $\mathsf{h}([\tau])$ returns a trivial discrete logarithm relation, then $\mathsf{e}(\overline{[\tau]})$ returns a valid witness.

Lemma 9. *Let τ, as shown in (16), be an accepting transcript of* ACSPf *such that $\tau \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{ACSPf}}$. If $\mathsf{h}([\tau])$ returns $(\mathbf{0}^n, \mathbf{0}^n, 0, 0, 0)$ then $\mathsf{e}(\overline{[\tau]})$ 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 $\mathsf{e}(\overline{[\tau]})$ fails to extract a valid witness for an accepting transcript $\tau \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{ACSPf}}$, $\mathsf{h}([\tau])$ outputs a non-trivial discrete logarithm relation, i.e., $\mathcal{H}$ succeeds. So we have that

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

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

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

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

$$e_{\mathbf{g}}^{(1)} = \mathbf{0}^n, e_{\mathbf{h}}^{(1)} = \mathbf{0}^n, e_u^{(1)} = 0, e_g^{(1)} = 0, e_h^{(1)} = 0, e_{\mathbf{g}}^{(2)} = \mathbf{0}^n, e_{\mathbf{h}}^{(2)} = \mathbf{0}^n, e_u^{(2)} = 0, e_g^{(2)} = 0, e_h^{(2)} = 0 .$$

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

$$\hat{t} = x^2(\delta(y,z) + \langle \mathbf{z}_{[1:]}^{Q+1}, \mathbf{c}\rangle) + t_{1g}x + \sum_{i=3}^{6} t_{ig}x^i . \tag{17}$$

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

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

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

$$\Big(\sum_{i=1}^{\log n} (l_{ig_{1+k}} x_i^2 + r_{ig_{1+k}} x_i^{-2}) + p'_{g_{1+k}}\Big) - a \cdot \left(\prod_{i=1}^{\log n} x_i^{(-1)^{1-\mathsf{bit}(k,i,\log n)}}\right) = 0 . \tag{18}$$

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

$$\Big(\sum_{i=1}^{\log n} (l_{ih_{1+k}} x_i^2 + r_{ih_{1+k}} x_i^{-2}) + p'_{h_{1+k}}\Big) - by^{(-(k))} \cdot \left(\prod_{i=1}^{\log n} x_i^{(-1)^{\mathsf{bit}(k,i,\log n)}}\right) = 0 . \tag{19}$$

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

$$\left(\sum_{i=1}^{\log n} (l_{iu} x_i^2 + r_{iu} x_i^{-2})\right) + p'_u - o \cdot ab = 0 . \tag{20}$$

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 $o \cdot \langle p'_{\mathbf{g}}, p'_{\mathbf{h}} \circ \mathbf{y}^n\rangle = p'_u$. 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/o. 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 (16), be an accepting transcript of* ACSPf *such that $\tau \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{ACSPf}}$. Let*

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

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

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

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

Additionally,

$$\left(\sum_{i=1}^{\log n} (l_{iu} x_i^2 + r_{iu} x_i^{-2}) \right) + p'_u - o \cdot ab = 0 \,. \tag{21}$$

Then

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

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

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

□

Since τ is an accepting transcript of ACSPf and $\tau \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{ACSPf}}$ and (18) to (20) hold, using Lemma 10, we get

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

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

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

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

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

Plugging in the value of $\hat{t}$ obtained in (17), we have that

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

Since $\tau \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{ACSPf}}$, 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.,

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

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

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

Simplifying and rearranging we get

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

Since $\tau \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{ACSPf}}$, 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 $p'_{\mathbf{g}}, p'_{\mathbf{h}}, y, p'_u, o$ 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 $p'_{\mathbf{g}}, p'_{\mathbf{h}}, y, p'_u, o$.

Proof (Lemma 8).

First we observe that given that $\mathsf{Bad}(\mathsf{params}, x, j) \neq \texttt{true}$, 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 $j \in \{0, \ldots, n-1\}$

$$\begin{aligned} f^{\mathbf{g}}_{k,j}(X) &= l_{kg_{1+j}} X^2 + r_{kg_{1+j}} X^{-2} + p'_{g_{1+j}} + \sum_{i=1}^{k-1} (l_{ig_{1+j}} x_i^2 + r_{ig_{1+j}} x_i^{-2}) \ , \\ f^{\mathbf{h}}_{k,j}(X) &= l_{kh_{1+j}} X^2 + r_{kh_{1+j}} X^{-2} + p'_{h_{1+j}} + \sum_{i=1}^{k-1} (l_{ih_{1+j}} x_i^2 + r_{ih_{1+j}} x_i^{-2}) \ . \end{aligned} \tag{22}$$

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

$$f^u_k(X) = l_{ku} X^2 + r_{ku} X^{-2} + p'_u + \sum_{i=1}^{k-1} (l_{iu} x_i^2 + r_{iu} x_i^{-2}) \ . \tag{23}$$

Using our notation in (22) and (23), 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.

$$f^u_{\log n}(x_{\log n}) - o \cdot ab = 0 \ .$$

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

$$f^u_{\log n}(x_{\log n}) - o \cdot f^{\mathbf{g}}_{\log n, 1}(x_{\log n}) \cdot f^{\mathbf{h}}_{\log n, 1}(x_{\log n}) = 0 \ . \tag{24}$$

In order to eliminate all variable except $p'_{\mathbf{g}}, p'_{\mathbf{h}}, y, p'_u, o$, we need to use the first two sets of equalities to obtain relations that we can plug back into (24).

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

$$a = f^{\mathbf{g}}_{\log n,k}(x_{\log n}) \cdot \left(\prod_{i=1}^{\log n} x_i^{(-1)^{\mathsf{bit}(k,i,\log n)}} \right) . \tag{25}$$

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

$$f^{\mathbf{g}}_{\log n,j}(x_{\log n}) \cdot x_t^2 = f^{\mathbf{g}}_{\log n,j+n/2^t}(x_{\log n}) . \tag{26}$$

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

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

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 $j \in \{0, \ldots, n/2^{t'} - 1\}$.

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

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

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

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

$$l_{tg_{1+j}} = 0 \ , r_{tg_{1+j}} = f^{\mathbf{g}}_{t-1,j+n/2^t}(x_{t-1}) \ , \tag{27}$$

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

$$r_{1g_{1+j}} = p'_{g_{1+j+n/2}} \ , l_{1g_{1+j}} = 0 \ . \tag{28}$$

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

$$r_{th_{1+j}} = 0 \ , l_{th_{1+j}} = f^{\mathbf{h}}_{t-1,j+n/2^t}(x_t) \cdot y^{n/2^t} \ . \tag{29}$$

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

$$r_{1h_{1+j}} = 0 \ , l_{1h_{1+j}} = p'_{h_{1+j+n/2}} \cdot y^{n/2} \ . \tag{30}$$

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

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

The base case for $k = \log n$ is true since (24) 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

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

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

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

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

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

From (27) and (29) 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

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

This can be simplified to

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

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

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

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

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

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

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

From (28) and (30) we have that for $j \in \{0, \dots, n/2 - 1\}$

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

So, we have that

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

Simplifying we get that

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

□

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). We consider a variant of Sonic that does not use the signature of correct computation.

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 $\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O \in \mathbb{Z}_p^{Q \times n}$.

The argument SnACSPf is again an argument of knowledge for the relation (15).

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 $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

DESCRIPTION OF SnACSPf. The setup algorithm SnACSPf.Setup on input 1^λ fixes integers n, d such that $3n < d < 4n$. It generates the generators g, h of the group $\mathbb{G}$ such that $e(g, h)$ is a generator of $\mathbb{G}_T$ and sets $\mathsf{bp} = (p, \mathbb{G}, \mathbb{G}_T, e, g, h)$. 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 finally returns $(n, d, \mathsf{bp}, \mathsf{srs})$. The prover and the verifier algorithms, SnACSPf.P, SnACSPf.V are shown in Figure 16.

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.[5] 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.

[5] 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.

```
$\mathsf{SnACSPf.P}(((\mathsf{bp}, \mathsf{srs}, n, d), (\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c})),$
        $(\mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O))$
$c_{n+1}, c_{n+2}, c_{n+3}, c_{n+4} \leftarrow_\$ \mathbb{Z}_p$
$r(X,Y) \leftarrow \langle \mathbf{a}_L, \mathbf{X}^{n+1}_{[1:]} \circ \mathbf{Y}^{n+1}_{[1:]} \rangle$
        $+\langle \mathbf{a}_R, \mathbf{X}^{-n-1}_{[1:]} \circ \mathbf{Y}^{n+1}_{[1:]} \rangle$
        $+\langle \mathbf{a}_O, \mathbf{X}^{-2n-1}_{[n+1:]} \circ \mathbf{Y}^{-2n-1}_{[n+1:]} \rangle$
        $+ \sum_{i=1}^{4} c_{n+i} X^{-2n-i} Y^{-2n-i}$
$R \leftarrow g^{\alpha x^{d-n} r(x,1)}$
                                        $\xrightarrow{R}$
                                        $\xleftarrow{y}$
                                                    $\mathsf{SnACSPf.V}((\mathsf{bp}, \mathsf{srs}, n, d), (\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c}))$
                                                    $y \leftarrow_\$ \mathbb{Z}_p^*$
$k(Y) \leftarrow \langle \mathbf{c}, \mathbf{Y}^{Q+n+1}_{[n+1:]} \rangle$
$s(X,Y) \leftarrow \mathbf{Y}^{Q+n+1}_{[n+1:]} \cdot (\mathbf{W}_L \cdot \mathbf{X}^{-n-1}_{[1:]} + \mathbf{W}_R \cdot \mathbf{X}^{n+1}_{[1:]}$
        $+\mathbf{W}_O \cdot \mathbf{X}^{2n+1}_{[n+1:]})$
        $+\langle -\mathbf{Y}^{n+1}_{[1:]} - \mathbf{Y}^{-n-1}_{[1:]}, \mathbf{X}^{2n+1}_{[n+1:]} \rangle$
                                                    $k(Y) \leftarrow \langle \mathbf{c}, \mathbf{Y}^{Q+n+1}_{[n+1:]} \rangle$
                                                    $s(X,Y) \leftarrow \mathbf{Y}^{Q+n+1}_{[n+1:]} \cdot (\mathbf{W}_L \cdot \mathbf{X}^{-n-1}_{[1:]}$
                                                            $+\mathbf{W}_R \cdot \mathbf{X}^{n+1}_{[1:]} + \mathbf{W}_O \cdot \mathbf{X}^{2n+1}_{[n+1:]})$
                                                            $+\langle -\mathbf{Y}^{n+1}_{[1:]} - \mathbf{Y}^{-n-1}_{[1:]}, \mathbf{X}^{2n+1}_{[n+1:]} \rangle$
$r'(X,Y) \leftarrow r(X,Y) + s(X,Y)$
$t(X,Y) \leftarrow r(X,1)r'(X,Y) - k(Y)$
$T \leftarrow g^{\alpha t(x,y)}$
                                        $\xrightarrow{T}$
                                        $\xleftarrow{z}$
                                                    $z \leftarrow_\$ \mathbb{Z}_p^*$
$a \leftarrow r(z,1)$
$w_a(X) \leftarrow \frac{r(X,1)-r(z,1)}{X-z}$
$W_a \leftarrow g^{w_a(x)}$
$b \leftarrow r(z,y)$
$w_b(X) \leftarrow \frac{r(X,1)-r(yz,1)}{X-yz}$
$W_b \leftarrow g^{w_b(x)}$
$t \leftarrow t(z,y)$
$w_t(X) \leftarrow \frac{t(X,y)-t(z,y)}{X-z}$
$W_t \leftarrow g^{w_t(x)}$
                                        $\xrightarrow{a, W_a, b, W_b, W_t}$
                                                    $t \leftarrow a(b + s(z,y)) - k(y)$
                                                    If $e(W_a, h^{\alpha x})e(g^a W_a^z, h^\alpha) \neq e(R, h^{x^{-d+n}})$ then
                                                      Return 0
                                                    If $e(W_b, h^{\alpha x})e(g^b W_b^{yz}, h^\alpha) \neq e(R, h^{x^{-d+n}})$ then
                                                      Return 0
                                                    If $e(W_t, h^{\alpha x})e(g^t W_t^z, h^\alpha) \neq e(T, h)$ then
                                                      Return 0
                                                    Return 1
```

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

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 (15). 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 the following corollary.

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 $\mathsf{SnACSPf} = \mathsf{SnACSPf}[\mathbb{G}, \mathbb{G}_T, e]$ be the interactive argument as described in Figure 16, for the relation R in (15). 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 $\lambda \in \mathbb{N}^+$*

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

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

Proof. (Theorem 7) We shall invoke Theorem 3 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 $\mathsf{CheckBad}$ 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 $\mathsf{SnACSPf}$ has two challenges, there are two definitions of $\mathsf{CheckBad}$ in Figure 17. 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 $\mathsf{CheckBad}$.

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

```
Procedure CheckBad([τ'], z):

$[\tau'] = (\mathsf{bp}, \mathsf{srs}, n, d, \mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c}; [R], y, [T])$
$k(Y) \leftarrow \langle \mathbf{c}, \mathbf{Y}^{Q+n+1}_{[n+1:]} \rangle$
$s(X,Y) \leftarrow \mathbf{Y}^{Q+n+1}_{[n+1:]} \cdot \mathbf{W}_L \cdot \mathbf{X}^{-n-1}_{[1:]} + \mathbf{Y}^{Q+n+1}_{[n+1:]} \cdot \mathbf{W}_R \cdot \mathbf{X}^{n+1}_{[1:]} + \mathbf{Y}^{Q+n+1}_{[n+1:]} \cdot \mathbf{W}_O \cdot \mathbf{X}^{2n+1}_{[n+1:]} + \langle -\mathbf{Y}^{n+1}_{[1:]} - \mathbf{Y}^{-n-1}_{[1:]}, \mathbf{X}^{2n+1}_{[n+1:]} \rangle$
$f(Z) \leftarrow \left( \sum_{\substack{i=-d \\ i \neq 0}}^{d} Z^i t_{g^{\alpha x^i}} \right) - \left( \sum_{\substack{i=n-2d \\ i \neq n-d}}^{n} Z^i r_{g^{\alpha x^{i-n+d}}} \right) \left( \sum_{\substack{i=n-2d \\ i \neq n-d}}^{n} (yZ)^i r_{g^{\alpha x^{i-n+d}}} - s(Z,y) \right) + k(y)$
Return $\mathsf{SZ}(f(Z), z)$

Procedure CheckBad([τ'], y):

$[\tau'] = (\mathsf{bp}, \mathsf{srs}, n, d, \mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c}; [R])$
For $i = 1, \ldots, n$ do
    $a^*_i \leftarrow r_{g^{\alpha x^{d-n+i}}}; b^*_i \leftarrow r_{g^{\alpha x^{d-n-i}}}; c^*_i \leftarrow r_{g^{\alpha x^{d-2n-i}}}$
$\mathbf{a}^*_L \leftarrow (a^*_1, \ldots, a^*_n); \mathbf{a}^*_R \leftarrow (b^*_1, \ldots, b^*_n); \mathbf{a}^*_O \leftarrow (c^*_1, \ldots, c^*_n)$
$f(Y) \leftarrow r_{g^{\alpha x^{d-n}}} r_{g^{\alpha x^{d-n}}} + \langle \mathbf{a}^*_L \circ \mathbf{a}^*_R - \mathbf{a}^*_O, \mathbf{Y}^{n+1}_{[1:]} + \mathbf{Y}^{-n-1}_{[1:]} \rangle + \mathbf{Y}^{Q+n+1}_{[n+1:]} \cdot (\mathbf{W}_L \cdot \mathbf{a}^*_L + \mathbf{W}_R \cdot \mathbf{a}^*_R + \mathbf{W}_O \cdot \mathbf{a}^*_O) - \langle \mathbf{c}, \mathbf{Y}^{Q+n+1}_{[n+1:]} \rangle$
Return $\mathsf{SZ}(f(Y), y)$
```

Fig. 17. The function CheckBad function for the SnACSPf.

```
Procedure e([τ̄]):

$\overline{[\tau]} = (\mathsf{bp}, \mathsf{srs}, n, d, \mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c}; [R], y, [T], z, (a, [W_a], b, [W_b], [W_t]))$
For $i = 1, \ldots, n$ do
    $a^*_i \leftarrow r_{g^{\alpha x^{d-n+i}}}; b^*_i \leftarrow r_{g^{\alpha x^{d-n-i}}}; c^*_i \leftarrow r_{g^{\alpha x^{d-2n-i}}}$
$\mathbf{a}^*_L \leftarrow (a^*_1, \ldots, a^*_n); \mathbf{a}^*_R \leftarrow (b^*_1, \ldots, b^*_n); \mathbf{a}^*_O \leftarrow (c^*_1, \ldots, c^*_n)$
Return $(\mathbf{a}^*_L, \mathbf{a}^*_R, \mathbf{a}^*_O)$
```

Fig. 18. The function e for SnACSPf.

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 $18n/(p-1)$. 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 18. It gets as input an extended accepting transcript $\overline{[\tau]}$ 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 3, 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.

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

$[\tau] = (\mathsf{bp}, \mathsf{srs}, n, d, \mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c}; [R], y, [T], z, (a, [W_a], b, [W_b], [W_t]))$
$k(Y) \leftarrow \langle \mathbf{c}, \mathbf{Y}^{Q+n+1}_{[n+1:]} \rangle$
$s(X,Y) \leftarrow \mathbf{Y}^{Q+n+1}_{[n+1:]} \cdot \mathbf{W}_L \cdot \mathbf{X}^{-n-1}_{[1:]} + \mathbf{Y}^{Q+n+1}_{[n+1:]} \cdot \mathbf{W}_R \cdot \mathbf{X}^{n+1}_{[1:]} + \mathbf{Y}^{Q+n+1}_{[n+1:]} \cdot \mathbf{W}_O \cdot \mathbf{X}^{2n+1}_{[n+1:]} \langle -\mathbf{Y}^{n+1}_{[1:]} - \mathbf{Y}^{-n-1}_{[1:]}, \mathbf{X}^{2n+1}_{[n+1:]} \rangle$
$f_1(X) \leftarrow (X-z) \left( \sum_{i=-d}^{d} X^i w_{ag^{x^i}} \right) + a - X^{-d+n} \left( \sum_{\substack{i=-d \\ i \neq 0}}^{d} X^i r_{g^{\alpha x^i}} \right)$
If $f_1(X) \neq 0$ then solve for $x^*$ such that $f_1(x^*) = 0$; If $g^x = g^{x^*}$ then return $x^*$
$f_2(X) \leftarrow (X-yz) \left( \sum_{i=-d}^{d} X^i w_{bg^{x^i}} \right) + b - X^{-d+n} \left( \sum_{\substack{i=-d \\ i \neq 0}}^{d} X^i r_{g^{\alpha x^i}} \right)$
If $f_2(X) \neq 0$ then solve for $x^*$ such that $f_2(x^*) = 0$; If $g^x = g^{x^*}$ then return $x^*$
$t \leftarrow a(b + s(z,y)) - k(y)$; $f_3(X) \leftarrow (X-z) \left( \sum_{i=-d}^{d} X^i w_{tg^{x^i}} \right) + t - \left( \sum_{\substack{i=-d \\ i \neq 0}}^{d} X^i t_{g^{\alpha x^i}} \right)$
If $f_3(X) \neq 0$ then solve for $x^*$ such that $f_3(x^*) = 0$; If $g^x = g^{x^*}$ then return $x^*$
Return $\perp$
```

Fig. 19. The function h_1 for SnACSPf.

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

$[\tau] = (\mathsf{bp}, \mathsf{srs}, n, d, \mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c}; [R], y, [T], z, (a, [W_a], b, [W_b], [W_t]))$
$k(Y) \leftarrow \langle \mathbf{c}, \mathbf{Y}^{Q+n+1}_{[n+1:]} \rangle$
$s(X,Y) \leftarrow \mathbf{Y}^{Q+n+1}_{[n+1:]} \cdot \mathbf{W}_L \cdot \mathbf{X}^{-n-1}_{[1:]} + \mathbf{Y}^{Q+n+1}_{[n+1:]} \cdot \mathbf{W}_R \cdot \mathbf{X}^{n+1}_{[1:]} + \mathbf{Y}^{Q+n+1}_{[n+1:]} \cdot \mathbf{W}_O \cdot \mathbf{X}^{2n+1}_{[n+1:]} + \langle -\mathbf{Y}^{n+1}_{[1:]} - \mathbf{Y}^{-n-1}_{[1:]}, \mathbf{X}^{2n+1}_{[n+1:]} \rangle$
$\mathsf{num}_1 \leftarrow -\alpha(x-z) \left( \sum_{i=-d}^{d} x^i w_{ag^{x^i}} + \sum_{\substack{i=-d \\ i \neq 0}}^{d} \alpha x^i w_{ag^{\alpha x^i}} \right) - \alpha a + x^{-d+n} \left( \sum_{i=-d}^{d} x^i r_{g^{x^i}} + \sum_{\substack{i=-d \\ i \neq 0}}^{d} \alpha x^i r_{g^{\alpha x^i}} \right)$
$\mathsf{den}_1 \leftarrow \alpha(x-z) \left( \sum_{i=-d}^{d} x^i w_{ah^{x^i}} + \alpha x^i w_{ah^{\alpha x^i}} \right) - x^{-d+n} \left( \sum_{i=-d}^{d} x^i r_{h^{x^i}} + \alpha x^i r_{h^{\alpha x^i}} \right)$
If $\mathsf{den}_1 \neq 0$ and $g^{\mathsf{num}_1/\mathsf{den}_1} = h$ then return $(\mathsf{num}_1/\mathsf{den}_1)$
$\mathsf{num}_2 \leftarrow -\alpha(x-yz) \left( \sum_{i=-d}^{d} x^i w_{bg^{x^i}} + \sum_{\substack{i=-d \\ i \neq 0}}^{d} \alpha x^i w_{bg^{\alpha x^i}} \right) - \alpha b + x^{-d+n} \left( \sum_{i=-d}^{d} x^i r_{g^{x^i}} + \sum_{\substack{i=-d \\ i \neq 0}}^{d} \alpha x^i r_{g^{\alpha x^i}} \right)$
$\mathsf{den}_2 \leftarrow \alpha(x-yz) \left( \sum_{i=-d}^{d} x^i w_{bh^{x^i}} + \alpha x^i w_{bh^{\alpha x^i}} \right) - x^{-d+n} \left( \sum_{i=-d}^{d} x^i r_{h^{x^i}} + \alpha x^i r_{h^{\alpha x^i}} \right)$
If $\mathsf{den}_2 \neq 0$ and $g^{\mathsf{num}_2/\mathsf{den}_2} = h$ then return $(\mathsf{num}_2/\mathsf{den}_2)$
$t \leftarrow a(b + s(z,y)) - k(y)$
$\mathsf{num}_3 \leftarrow -\alpha(x-z) \left( \sum_{i=-d}^{d} x^i w_{tg^{x^i}} + \sum_{\substack{i=-d \\ i \neq 0}}^{d} \alpha x^i w_{tg^{\alpha x^i}} \right) - \alpha t + \left( \sum_{i=-d}^{d} x^i t_{g^{x^i}} + \sum_{\substack{i=-d \\ i \neq 0}}^{d} \alpha x^i t_{g^{\alpha x^i}} \right)$
$\mathsf{den}_3 \leftarrow \alpha(x-z) \left( \sum_{i=-d}^{d} x^i w_{th^{x^i}} + \alpha x^i w_{th^{\alpha x^i}} \right) - \left( \sum_{i=-d}^{d} x^i t_{h^{x^i}} + \alpha x^i t_{h^{\alpha x^i}} \right)$
If $\mathsf{den}_3 \neq 0$ and $g^{\mathsf{num}_3/\mathsf{den}_3} = h$ then return $(\mathsf{num}_3/\mathsf{den}_3)$
Return $\perp$
```

Fig. 20. The function h_2 for SnACSPf.

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

$[\tau] = \left(\mathsf{bp}, \mathsf{srs}, n, d, \mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c}; [R], y, [T], z, (a, [W_a], b, [W_b], [W_t])\right)$
$k(Y) \leftarrow \langle \mathbf{c}, \mathbf{Y}^{Q+n+1}_{[n+1:]} \rangle$
$s(X,Y) \leftarrow \mathbf{Y}^{Q+n+1}_{[n+1:]} \cdot \mathbf{W}_L \cdot \mathbf{X}^{-n-1}_{[1:]} + \mathbf{Y}^{Q+n+1}_{[n+1:]} \cdot \mathbf{W}_R \cdot \mathbf{X}^{n+1}_{[1:]} + \mathbf{Y}^{Q+n+1}_{[n+1:]} \cdot \mathbf{W}_O \cdot \mathbf{X}^{2n+1}_{[n+1:]} + \langle -\mathbf{Y}^{n+1}_{[1:]} - \mathbf{Y}^{-n-1}_{[1:]}, \mathbf{X}^{2n+1}_{[n+1:]} \rangle$
$f_1(\mathrm{A}) \leftarrow \mathrm{A}(x - z)\left(\sum_{i=-d}^{d} x^i w_{a g^{x^i}} + \sum_{\substack{i=-d \\ i \neq 0}}^{d} \mathrm{A} x^i w_{a g^{\mathrm{A} x^i}}\right) + \mathrm{A}a - x^{-d+n}\left(\sum_{i=-d}^{d} x^i r_{g^{x^i}} + \sum_{\substack{i=-d \\ i \neq 0}}^{d} \mathrm{A} x^i r_{g^{\alpha x^i}}\right)$
If $f_1(\mathrm{A}) \neq 0$ then solve for $\alpha^*$ such that $f_1(\alpha^*) = 0$; If $g^\alpha = g^{\alpha^*}$ then return $\alpha^*$
$f_2(\mathrm{A}) \leftarrow \mathrm{A}(x - yz)\left(\sum_{i=-d}^{d} x^i w_{b g^{x^i}} + \sum_{\substack{i=-d \\ i \neq 0}}^{d} \mathrm{A} x^i w_{b g^{\mathrm{A} x^i}}\right) + \mathrm{A}b - x^{-d+n}\left(\sum_{i=-d}^{d} x^i r_{g^{x^i}} + \sum_{\substack{i=-d \\ i \neq 0}}^{d} \mathrm{A} x^i r_{g^{\alpha x^i}}\right)$
If $f_2(\mathrm{A}) \neq 0$ then solve for $\alpha^*$ such that $f_2(\alpha^*) = 0$; If $g^\alpha = g^{\alpha^*}$ then return $\alpha^*$
$t \leftarrow a(b + s(z, y)) - k(y)$
$f_3(\mathrm{A}) \leftarrow \mathrm{A}(x - z)\left(\sum_{i=-d}^{d} x^i w_{t g^{x^i}} + \sum_{\substack{i=-d \\ i \neq 0}}^{d} \mathrm{A} x^i w_{t g^{\mathrm{A} x^i}}\right) + \mathrm{A}t - \left(\sum_{i=-d}^{d} x^i r_{g^{x^i}} + \sum_{\substack{i=-d \\ i \neq 0}}^{d} \mathrm{A} x^i t_{g^{\alpha x^i}}\right)$
If $f_3(\mathrm{A}) \neq 0$ then solve for $\alpha^*$ such that $f_3(\alpha^*) = 0$; If $g^\alpha = g^{\alpha^*}$ then return $\alpha^*$
Return $\perp$
```

Fig. 21. The function h_3 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 $[\tau], \alpha, \beta$ and outputs whatever h_1 outputs. The function h_1 is defined in Figure 19.

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 $[\tau], x, \alpha$ and outputs whatever h_2 outputs. The function h_2 is defined in Figure 20.

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 $\mathsf{SnACSPf.Setup}$. 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 $[\tau], x, \beta$ and outputs whatever h_3 outputs. The function h_3 is defined in Figure 21.

We first 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 $O(n)$. The running time of $\mathcal{F}_1$ consists of the time required to answers q queries, run $\mathsf{SnACSPf.V}$ in at most q paths in the execution tree and the time required to run h_1. Hence its time complexity is $O(q \cdot n)$.
- 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 $O(n)$. The running time of $\mathcal{F}_2$ consists of the time required to answers q queries, run $\mathsf{SnACSPf.V}$ in at most q paths in the execution tree and the time required to run h_2. Hence its time complexity is $O(q \cdot n)$.
- 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 $O(n)$. The running time of $\mathcal{F}_3$ consists of the time required to answers q queries, run $\mathsf{SnACSPf.V}$ in at most q paths in the execution tree and the time required to run h_3. Hence its time complexity is $O(q \cdot n)$.

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

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

$$\tau = \left(\mathsf{bp}, \mathsf{srs}, n, d, \mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{c}; [R], y, [T], z, (a, [W_a], b, [W_b], [W_t])\right)$$

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

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

□

We shall next prove Lemma 11.

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

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

We therefore have

$$\begin{aligned} & \alpha(x-z)\left(\sum_{i=-d}^{d} x^i w_{ag^{x^i}} + \beta x^i w_{ah^{x^i}} + \alpha\beta x^i w_{ah^{\alpha x^i}} + \sum_{\substack{i=-d\\ i\neq 0}}^{d} \alpha x^i w_{ag^{\alpha x^i}}\right) \\ & + \alpha a - x^{-d+n}\left(\sum_{i=-d}^{d} x^i r_{g^{x^i}} + \beta x^i r_{h^{x^i}} + \alpha\beta x^i r_{h^{\alpha x^i}} + \sum_{\substack{i=-d\\ i\neq 0}}^{d} \alpha x^i r_{g^{\alpha x^i}}\right) = 0 \ . \end{aligned} \tag{31}$$

Consider the values $\mathsf{num}_1, \mathsf{den}_1$ in $\mathsf{h}_2([\tau], x, \alpha)$ – the above equation can be re-written as $-\mathsf{num}_1 + \beta\mathsf{den}_1 = 0$. Since $\mathsf{h}_2([\tau], x, \alpha)$ returned $\perp$ we have that den_1 must be 0, 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 (31) we get that

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

Therefore α is a root $f_1(\mathrm{A}) = 0$ in h_3. Since $\mathsf{h}_3([\tau], x, \beta)$ returned $\perp$ it means that $f_1(\mathrm{A})$ 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 $f_1(X) = 0$ in h_1. Now, since $\mathsf{h}_1([\tau], x, \beta)$ 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 $e(W_b, h^{\alpha x})e(g^b W_b^{yz}, h^{\alpha}) = e(R, h^{x^{-d+n}})$ and $e(W_t, h^{\alpha x})e(g^t W_t^{z}, h^{\alpha}) = e(T, h)$ hold. Using arguments similar to the ones we used above, we can show that

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

From the description of $\mathsf{SnACSPf.V}$, we have that

$$t = a(b + s(z,y)) - k(y) \; .$$

Plugging the values of a, b, t into the above equation 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 $\tau|_z \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{SnACSPf}}$, 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

$$\begin{aligned} & r_{g^{\alpha x^{d-n}}} r_{g^{\alpha x^{d-n}}} + \langle \mathbf{a}_L^* \circ \mathbf{a}_R^* - \mathbf{a}_O^*, \mathbf{y}_{[1:]}^{n+1} + \mathbf{y}_{[1:]}^{-n-1} \rangle \\ & + \mathbf{y}_{[n+1:]}^{Q+n+1} \cdot (\mathbf{W}_L \cdot \mathbf{a}_L^* + \mathbf{W}_R \cdot \mathbf{a}_R^* + \mathbf{W}_O \cdot \mathbf{a}_O^*) - \langle \mathbf{c}, \mathbf{y}_{[n+1:]}^{Q+n+1} \rangle = 0 \end{aligned}$$

Since $\tau|_y \notin \mathcal{T}_{\mathsf{BadCh}}^{\mathsf{SnACSPf}}$ 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

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

□

Acknowledgements.

We thank Joseph Jaeger for extensive discussions and his involvement in the earlier stages of this work. 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.

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*, November 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.

NRSW20. Jonas Nick, Tim Ruffing, Yannick Seurin, and Pieter Wuille. Musig-dn: Schnorr multi-signatures with verifiably deterministic nonces. *Cryptology ePrint Archive:2020/1057*, 2020.

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

Sch90. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, Heidelberg, August 1990.

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 Online extraction in the AGM: Limitation

We present an interactive proof system DS that does not allow for online extraction. We shall assume that $\mathsf{DS} = \mathsf{DS}[\mathbb{G}]$ is instantiated on an understood family of groups $\mathbb{G} = \{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}^+}$ of order $p = p(\lambda)$. It is a proof of knowledge of discrete logarithm of two group elements. More precisely, it is a proof of knowledge for the following relation

$$R = \left\{ (g, (X_1, X_2), (x_1, x_2)) : X_1 = g^{x_1} \wedge X_2 = g^{x_2} \right\} .$$

The setup algorithm DS.Setup on input 1^λ returns a generator g of the group $\mathbb{G}_\lambda$. The instance is a pair of group elements X_1, X_2 and the prover has witness x_1, x_2 which are the discrete logarithms of X_1, X_2 with respect to g. The prover and the verifier are formally described in Figure 22.

The soundness of DS can be argued in the standard model as follows. Given three accepting transcripts with the same first prover message, with high probability we can extract a witness (x_1^*, x_2^*) that satisfy $g^{x_1^*} = X_1$ and $g^{x_2^*} = X_2$. More concretely let τ_1, τ_2, τ_3 be three accepting transcripts such that $\tau_i = \{g, (X_1, X_2); A, (c_{i1}, c_{i2}), d_i\}$ for $i = 1, 2, 3$. Let x_1^*, x_2^* be the discrete logarithm for X_1, X_2. Since τ_1, τ_2, τ_3 are accepting transcripts, the following system of equation holds.

$$\begin{aligned} d_1 &= c_{11}x_1^* + c_{12}x_2^* + a \\ d_2 &= c_{21}x_1^* + c_{22}x_2^* + a \\ d_3 &= c_{31}x_1^* + c_{32}x_2^* + a \end{aligned}$$

$$\begin{array}{lcl}
\mathsf{DS.P}(((\mathbb{G}, g, p), (X_1, X_2)), (x_1, x_2)) & & \mathsf{DS.V}((\mathbb{G}, g, p), (X_1, X_2)) \\
a \leftarrow\!\!\$\ \mathbb{Z}_p; A \leftarrow g^a & \xrightarrow{A} & \\
 & \xleftarrow{c_1, c_2} & c_1, c_2 \leftarrow\!\!\$\ \mathbb{Z}_p \\
d \leftarrow a + c_1 x_1 + c_2 x_2 & \xrightarrow{d} & \text{Return } (g^d = X_1^{c_1} X_2^{c_2} A)
\end{array}$$

Fig. 22. Interactive proof system DS that does not allow for online extraction in the AGM.

The above system of equation has three variables x_1^*, x_2^*, a. Since c_{ij}'s are picked at random, with high probability the system of equations can be solved. Therefore, the protocol DS is sound in the standard model.

Next we argue that online extraction is not possible for DS in the AGM. In particular, we look at an accepting transcript produced by an honest prover and argue that we cannot extract a witness from the transcript. Given an accepting transcript $\tau = \{g, (X_1, X_2); A, (c_1, c_2), d\}$, using the representation of A with respect to g (an honest prover must have provided the representation of A only in terms of g), we can write

$$d = c_1 x_1^* + c_2 x_2^* + a_g \ .$$

We have two variables x_1^*, x_2^* and just one equation. So, we cannot extract x_1^*, x_2^* just from one accepting transcript, i.e., DS does not support online extraction in the AGM.

One reason why online extraction in the AGM was not possible for this protocol while it was possible for $\mathsf{RngPf}, \mathsf{ACSPf}$ is that here x_1^*, x_2^* were committed in the exponent of the same generator – this leads to us not having sufficient number of equations to solve for x_1^*, x_2^* from the single transcript. For similar reasons, online extraction is not possible for the generalized version of ACSPf that supports additional commitments and the aggregated range proofs in [BBB+18].