Revisiting BBS Signatures*

Stefano Tessaro and Chenzhi Zhu

Paul G. Allen School of Computer Science & Engineering
University of Washington, Seattle, US
{tessaro,zhucz20}@cs.washington.edu

Abstract. BBS signatures were implicitly proposed by Boneh, Boyen, and Shacham (CRYPTO '04) as part of their group signature scheme, and explicitly cast as stand-alone signatures by Camenisch and Lysyanskaya (CRYPTO '04). A provably secure version, called BBS+, was then devised by Au, Susilo, and Mu (SCN '06), and is currently the object of a standardization effort which has led to a recent RFC draft. BBS+ signatures are suitable for use within anonymous credential and DAA systems, as their algebraic structure enables efficient proofs of knowledge of message-signature pairs that support partial disclosure.

BBS+ signatures consist of one group element and two scalars. As our first contribution, we prove that a variant of BBS+ producing shorter signatures, consisting only of one group element and one scalar, is also secure. The resulting scheme is essentially the original BBS proposal, which was lacking a proof of security. Here we show it satisfies, under the q-SDH assumption, the same provable security guarantees as BBS+. We also provide a complementary tight analysis in the algebraic group model, which heuristically justifies instantiations with potentially shorter signatures.

Furthermore, we devise simplified and shorter zero-knowledge proofs of knowledge of a BBS message-signature pair that support partial disclosure of the message. Over the BLS12-381 curve, our proofs are 896 bits shorter than the prior proposal by Camenisch, Drijvers, and Lehmann (TRUST '16), which is also adopted by the RFC draft.

Finally, we show that BBS satisfies one-more unforgeability in the algebraic group model in a scenario, arising in the context of credentials, where the signer can be asked to sign arbitrary group elements, meant to be commitments, without seeing their openings.

1 Introduction

The seminal works of Camenisch and Lysyanskaya [CL03, CL04] highlighted how certain digital signature schemes with suitable algebraic structures are amenable to applications such as anonymous credentials, direct anonymous attestation (DAA), and group signatures. These schemes easily enable the signing of a *commitment*, typically by being algebraically compatible with a Pedersen commitment [Ped92], and support very efficient zero-knowledge proofs of knowledge of a valid message-signature pair.

This paper revisits and improves BBS signatures [BBS04, ASM06, CDL16], one of the most efficient pairing-based schemes with these properties, which has recently been in the midst of renewed interest in the context of decentralized identity. This has led to reference implementations [BBSa, BBSb], to a standardization effort by the W3C Verifiable Credentials Working group, and to an RFC draft [LKWL22]. BBS is also a building block for DAA [Che09, BL10, CDL16], and is used by Intel SGX's EPID protocol [BL11]. Furthermore, BBS signatures are theoretically interesting, due to their simplicity and efficiency. Most applications, and the RFC draft, consider the provably-secure version of BBS referred to as BBS+ [CDL16, ASM06], whose signatures consist of *one* group element in $\mathbb{G}_1$ and *two* scalars in $\mathbb{Z}_p$, where p is the group order.

* An extended abstract of this paper appears at EUROCRYPT 2023. This is the full version.

Our contributions, in a nutshell. As our first main contribution, we prove the strong unforgeability of a variant of BBS+ which produces shorter signatures only consisting of one group element and one scalar. The resulting scheme is in fact the original BBS signature scheme by Boneh, Boyen, and Shacham [BBS04] as stated by Camenisch and Lysyanskaya [CL04], for which however no proof of security was known. Our new proof gives us a more efficient version of the scheme that can replace BBS+ in applications and standards with no loss, and re-affirms the security of prior works (e.g., [Che09, BL10]) which already used this optimized version but relied on an incorrect proof.

Furthermore, we provide a tighter security proof in the Algebraic Group Model [FKL18], which also supports potentially shorter signatures. We also optimize the associated proofs of knowledge of BBS signatures, achieving substantial savings over the current state-of-the-art [LKWL22]. Finally, we study the security of BBS in contexts where group elements are signed, and show that the scheme satisfies, in the AGM, a security property which is a natural weakening of what is achieved by structure-preserving signatures [AFG+10].

BBS+. The BBS+ scheme was proposed by Au, Susilo, and Mu [ASM06], based on ideas by [BBS04, CL04], and proved secure under the q-SDH assumption. The proof was then adapted to type-3 pairings by Camenisch, Drijvers, and Lehmann [CDL16]. It signs vectors $\boldsymbol{m} \in \mathbb{Z}_p^\ell$. To do so, the public parameters consist of $\ell + 2$ generators $g_1, h_0, h_1, \ldots, h_\ell \in \mathbb{G}_1$, and a signature has the format (A, e, s), where $s, e \in \mathbb{Z}_p$ are randomly chosen, and

$$A = \left(g_1 h_0^s \prod_{i=1}^{\ell} h_i^{\boldsymbol{m}[i]} \right)^{\frac{1}{x+e}} .$$

Here, $x \in \mathbb{Z}_p$ is the secret key, and given the public key $X_2 = g_2^x \in \mathbb{G}_2$, and a pairing, it is easy to verify a valid BBS+ signature.

Security for BBS signatures. The only difference between BBS and BBS+ is the additional term h_0^s in the latter, which mandates the inclusion of s in the signature. A natural question is whether this term is necessary, or instead an artifact of the proofs [ASM06, CDL16]. Indeed, no attack seems to affect plain BBS, without the term h_0^s, but prior proof attempts (e.g., [BL10]) contained fundamental errors.

We prove that (plain) BBS signatures are indeed secure under the q-SDH assumption. The concrete security guarantees are essentially identical to those established for BBS+, and this suggests a more efficient drop-in replacement for BBS+ in existing applications. Our techniques close in particular gaps left by incorrect proofs, and can be used to prove exculpability of the original BBS group signature scheme [BBS04].

Tight AGM bounds. Our new proof, not unlike the prior proofs for BBS+, is not tight, i.e., it incurs a multiplicative loss equal to the number of signing queries q. As a strong hint that this loss may be artificial, we give a tight proof for BBS signatures in the Algebraic Group Model [FKL18].

Our AGM analysis also addresses a different artificial aspect of the standard-model analysis, namely the random choice of e values from $\mathbb{Z}_p$. Instead, our AGM analysis merely asks that these values are unlikely to collide, and their collision probability becomes a term (meant to be negligible) in the security bound. This allows for more flexibility, in that the e values could be generated from a counter or (assuming random oracles) as a hash of the message. It also suggests a BBS variant, which we call *truncated BBS*, where e is chosen from $\mathbb{Z}_{2^{2\lambda}}$, where λ is the desired security level (typically, $\lambda = 128$). On BLS12-381, this does not have any benefit. However, as in all schemes

based on q-SDH, one may want to assess the potential impact of attacks such as those by Brown and Gallant [BG04] and Cheon [Che06] and choose an even bigger curve—in that case, truncation of the scalar may become effective.

Signing commitments. An important question is to which extent BBS can be thought as a signature scheme signing a user-supplied group element, i.e., an element $C \in \mathbb{G}_1$ is signed as $(C^{\frac{1}{x+e}}, e)$. Indeed, in the context of blind issuance of credentials, one can think of $C = g_1 h_1^{m_1} \cdots h_\ell^{m_\ell}$ as a commitment sent from the user to the signer, and the signer's response $(C^{1/(x+e)}, e)$ is a valid BBS signature on $\boldsymbol{m} = (m_1, \ldots, m_\ell)$. It is not hard to see that this form of BBS does not yield a secure signature scheme over group elements as e.g., a signature on C can easily be transformed, by squaring it, into a signature on C^2. However, we show that, in the AGM, BBS satisfies a form of one-more unforgeability, where obtaining signatures of q group elements does not enable the attacker to produce valid BBS signatures (e.g., by "opening" these group elements as commitments) on more than q messages. This is sufficient in the context of blind issuance.

Zero-knowledge PoKs. We also revisit the problem of proving knowledge of a BBS message-signature pair with new zero-knowledge proofs of knowledge which are shorter than state-of-the art solutions adopted in the RFC draft [LKWL22] and initially proposed in [CDL16]. To prove knowledge of a BBS message-signature pair $(\boldsymbol{m}, \sigma)$, without revealing k out of the ℓ components of $\boldsymbol{m}$, our proof (when compiled as a NIZK via the Fiat-Shamir transform) consists of 2 elements in $\mathbb{G}_1$, as well as $k + 3$ scalars in $\mathbb{Z}_p$. The proof adopted in the RFC draft, in contrast, consists of 3 elements in $\mathbb{G}_1$, and $k + 5$ scalars. While a reduction by one scalar is possible due to our removal of the random value s from a signature, the remaining optimizations are the result of a different approach.

Related schemes. We note that when signing individual elements of $\mathbb{Z}_p$, the simpler Boneh-Boyen signatures [BB08] would typically outperform BBS. The closest scheme to BBS is the one by Pointcheval-Sanders (PS) [PS16]. PS signatures consist of two group elements, and are comparably efficient to the short version of BBS from this paper. However, both the public and the secret keys grow linearly with ℓ, the length of the message vector to be signed, whereas in BBS they consist of a single element. (The group generators in BBS can be generated as the output of a hash function, and they should not be part of the key materials.) PS signatures feature properties which BBS does not possess, including re-randomizability and aggregability. The latter property is essential for their use in the recent Coconut system [SAB+19], for which BBS does not appear suitable.

Outline. Our new proof for BBS is given in Section 3, followed by our AGM analysis in Section 4. Our new zero-knowledge proofs are given in Section 5, and our analysis of BBS as a signature scheme on group elements is in Section 6. We give a technical overview next.

1.1 Technical Overview

New BBS proof. It is instructive to first review existing proofs [ASM06, CDL16] for BBS+. To this end, we consider the special case where we sign a *single* scalar $m \in \mathbb{Z}_p$, i.e., a signature under secret key $x \in \mathbb{Z}_p$ takes form, for random $s, e \in \mathbb{Z}_p$,

$$\sigma = (A, s, e) \ , \quad \text{where} \quad A = (g_1 h_0^s h_1^m)^{\frac{1}{x+e}} \ .$$

If an attacker obtains q adaptively chosen signatures (A_i, s_i, e_i) for message $m_i \in \mathbb{Z}_p$ and finally produces a valid forgery (A^*, e^*, s^*) for a message $m^* \in \mathbb{Z}_p$, we can identify three cases, which are to be addressed differently:

(1) There exists an $i \in [q]$ such that $A_i = A^*$ and $e_i = e^*$
(2) There exists an $i \in [q]$ such that $A_i \neq A^*$ and $e_i = e^*$
(3) $e^* \notin \{e_1, \ldots, e_q\}$.

The most challenging case is **(2)**. Indeed, **(1)** implies that $g_1 h_0^{s_i} h_1^{m_i} = g_1 h_0^{s^*} h_1^{m^*}$, while $(s_i, m_i) \neq (s^*, m^*)$, which in turn implies a break of the discrete logarithm assumption in $\mathbb{G}_1$. For **(3)**, instead, one resorts to a by-now classical technique by Boneh and Boyen [BB08]. The key point here is that to break q-SDH, given $g_1, g_1^x, \ldots, g_1^{x^q} \in \mathbb{G}_1$, along with g_2, g_2^x in $\mathbb{G}_2$, it is enough to compute $g_1^{p(x)/(x+e)}$, for a polynomial $p(\mathsf{X})$ which is not divisible by $\mathsf{X} + e$. This indeed allows us to recover $g_1^{1/(x+e)}$, which gives us a valid q-SDH solution.

To do so, the reduction picks $e_1, \ldots, e_q$ ahead of time. It uses g_2^x as the public key, but uses new generators $\overline{g}_1 = g_1^{\theta p(x)}$, $h_0 = g_1^{\alpha p(x)}$ and $h_1 = g_1^{\beta p(x)}$ for $\mathbb{G}_1$, where $p(\mathsf{X}) = \prod_{i=1}^{q} (\mathsf{X} + e_i)$ and $\alpha, \beta, \theta \leftarrow\!\!\$\, \mathbb{Z}_p$. Note that $\overline{g}_1, h_0, h_1$ can easily be computed from $g_1^{x^i}$ for $i \in [q]$, since $p(\mathsf{X})$ has degree q, and that for any m, s, and $i \in [q]$, the reduction can always simulate a signature $(\overline{g}_1 h_0^s h_1^m)^{\frac{1}{x+e_i}}$, since $p(\mathsf{X})$ is divisible by $\mathsf{X} + e_i$. Moreover, a forgery for $e^* \notin \{e_1, \ldots, e_q\}$, allows us to compute $(e^*, g_1^{\frac{1}{x+e^*}})$, and break q-SDH.

Handling Case (2). The value s was crucial in [ASM06, CDL16] to deal with **(2)**. To see how it was used, let us assume that we can actually guess the index i for which **(2)** occurs. Then, with $p'(\mathsf{X}) = \prod_{j \neq i} (\mathsf{X} + e_j)$, the reduction can set

$$\overline{g}_1 = g_1^{\theta p'(x)} \;, \quad h_0 = \overline{g}_1^{\frac{(x+e_i)\delta - 1}{\alpha}} \;, \quad h_1 = h_0^{\beta} \;.$$

Queries for $j \neq i$ can be answered as above for any (s, m), since $p'(\mathsf{X})$ is divisible by $\mathsf{X} + e_i$. In contrast, for the i-th query, on message m_i, the reduction can only answer for the specific choice of $s_i = \alpha - \beta \cdot m_i$, since

$$A_i = (\overline{g}_1 h_0^{s_i} h_1^{m_i})^{\frac{1}{x+e_i}} = \left(\overline{g}_1^{(x+e_i)\delta}\right)^{\frac{1}{x+e_i}} = \overline{g}_1^{\delta} \;.$$

Despite the fact that s_i depends on m_i, one can show that its distribution is uniform. If the attacker now produces a forgery with $e^* = e_i$, it means that we have

$$A^* = \left(\overline{g}_1 h_0^{s^*} h_1^{m^*}\right)^{\frac{1}{x+e_i}} = \overline{g}_1^{\frac{1}{x+e_i}\left(1 + \frac{(x+e_i)\delta - 1}{\alpha}(s^* + \beta m^*)\right)}$$

and the reduction can solve q-SDH because $1 + \frac{(\mathsf{X}+e_i)\delta - 1}{\alpha}(s^* + \beta m^*)$ is not divisible by $\mathsf{X} + e_i$.

Our improvement. The reduction for BBS+ programs s in a message-dependent way to handle **(2)**. Our main idea here is to let e play this role instead, thus dispensing with the use of s, and in fact obtaining a slightly simpler reduction. Concretely, for BBS, we drop h_0, as it is not needed any more, and we set up

$$\overline{g}_1 = g_1^{\alpha p'(x)(x+\varepsilon_i)} \;, \quad h_1 = g_1^{\beta p'(x)} \;.$$

Here, $p'(\mathsf{X})$ is as above, and $\alpha, \beta, \varepsilon_i$ are random, and, most importantly, ε_i will not necessarily equal e_i. Now, every query $j \neq i$ can be answered as before. To answer the i-th query, however, we first observe that

$$C_i = \overline{g}_1 h_1^{m_i} = g_1^{\alpha p'(x)(x+\varepsilon_i)} \cdot g_1^{\beta p'(x) m_i} = g_1^{\alpha p'(x)(x + \varepsilon_i + \frac{\beta}{\alpha} m_i)} \;.$$

We are going to show that if we set $e_i = \varepsilon_i + \frac{\beta}{\alpha} m_i$, not only we can compute $A_i = C_i^{\frac{1}{x+e_i}}$, but *also*, this e_i has the right distribution. This last argument is somewhat involved. For example, it turns out that if message m_i is such that $x + \varepsilon_i + \frac{\beta}{\alpha} m_i = 0$, the distribution of e_i is not correct. Luckily, however, this is the only case, and moreover, if this indeed happened, this would mean $x = -\varepsilon_i - \frac{\beta}{\alpha} m_i$, and we could break q-SDH directly. If we then obtain a forgery (A^*, e^*) with $e^* = e_i$ for a message m^*, we note that the discrete logarithm of A^* is

$$\mathrm{DL}_{g_1}(A^*) = \frac{\alpha p'(x)(x + \varepsilon_i + \frac{\beta}{\alpha} m^*)}{x + \varepsilon_i + \frac{\beta}{\alpha} m_i} .$$

However, one can show that $\mathsf{X} + \varepsilon_i + \frac{\beta}{\alpha} m_i$ does not divide $\alpha p'(\mathsf{X})(\mathsf{X} + \varepsilon_i + \frac{\beta}{\alpha} m^*)$ if $m_i \neq m^*$.

AGM Security. In the AGM, we restrict our focus to algebraic adversaries, and in the case of BBS, this means that the adversary outputs a forgery A^* and its representation in terms of $g_1, h_1, \ldots, h_\ell$, as well as the $\mathbb{G}_1$-part the prior signatures $A_1, \ldots, A_q$. We note that the discrete logarithm of each A_i equals $\varphi^{\boldsymbol{y}}_{\boldsymbol{m}_i, e_i}(x)$, where x is the secret key, $\boldsymbol{y}$ is the vectors of discrete logarithms of $\boldsymbol{h}_1$ relative to g_1, and

$$\varphi^{\boldsymbol{y}}_{\boldsymbol{m}_i, e_i}(\mathsf{X}) = \frac{1 + \langle \boldsymbol{y}, \boldsymbol{m}_i \rangle}{\mathsf{X} + e_i} .$$

Analogously, the discrete logarithm of A^* for a forgery A^*, e^* for a message $\boldsymbol{m}^*$ equals $\varphi^{\boldsymbol{y}}_{\boldsymbol{m}^*, e^*}(x)$. Further, the algebraic adversary gives us a representation of $\varphi^{\boldsymbol{y}}_{\boldsymbol{m}^*, e^*}(x)$ as an affine combination of the $\varphi^{\boldsymbol{y}}_{\boldsymbol{m}_i, e_i}(x)$'s. Our key observation is that unless some very specific properties are satisfied by $\boldsymbol{y}$, the *function* $\varphi^{\boldsymbol{y}}_{\boldsymbol{m}^*, e^*}(\mathsf{X})$, as opposed to its evaluation at x, *cannot* be expressed as an affine combination of the functions $\varphi^{\boldsymbol{y}}_{\boldsymbol{m}_i, e_i}(\mathsf{X})$. Therefore, x must be a zero to a (known) polynomial of degree at most q, and it can be recovered by factoring this polynomial. It also turns out that whenever the choice of $\boldsymbol{y}$ does not allow this argument to go through, we are going to be able to recover a non-trivial discrete-logarithm relation, and break the discrete logarithm problem directly.

BBS Signatures of Committed Values. Our AGM proof will enable us to also study the scenario where the adversary can query an oracle on an arbitrary $C \in \mathbb{G}_1$ and obtain $C^{\frac{1}{x+e}}$ for a random e. We show that in the AGM it is impossible, except with negligible probability, to come up with $q + 1$ valid BBS signatures upon querying this oracle q times. The main difficulty is that an AGM adversary here can query this oracle with group elements which are combinations of the outputs of previous queries. However, we are going to show how an algebraic adversary making q oracle queries can be simulated by one making queries to the actual BBS signing oracle. To do this, we rely on a property of our AGM proof above, namely that the statement holds even if the values $e_1, e_2, \ldots$ are known to the adversary beforehand, and we use this to give an inductive argument which shows how to build these signing queries in order to emulate the oracle signing a group element instead.

New Proofs of Knowledge. We give new Σ-protocols to prove knowledge of a message-signature pair for BBS, given, possibly, partial knowledge of the message. Our basic observation is that valid signature (A, e) for $\boldsymbol{m}$ satisfies $\mathsf{e}(A, X_2) = \mathsf{e}(B, g_2)$, where $B = C(\boldsymbol{m})A^{-e}$, and $C(\boldsymbol{m}) = g_1 \prod_{i=1}^{\ell} h_i^{\boldsymbol{m}[i]}$. For the case where $\boldsymbol{m}$ is fully known to the verifier, for example, our prover commits to a randomized version of A, B, namely $\overline{A} = A^r$ and $\overline{B} = B^r = C(\boldsymbol{m})^r \overline{A}^{-e}$. Then, the prover engages in a homomorphism proof [Mau09] to show knowledge of a representation (r, e) of $\overline{B}$ to the base $C(\boldsymbol{m})$ and $\overline{A}$.

Game $\mathsf{SUF}^{\mathcal{A}}_{\mathsf{SS}}(\lambda)$:	Oracle $\textsc{Sign}(M)$:
$\mathsf{S} \leftarrow \emptyset$, $\mathsf{par} \leftarrow\!\!\$\ \mathsf{SS.Setup}(1^\lambda)$	$\sigma \leftarrow\!\!\$\ \mathsf{SS.Sign}(\mathsf{par}, \mathsf{sk}, M)$
$(\mathsf{vk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{SS.KG}(\mathsf{par})$	If $\sigma \neq \bot$ then $\mathsf{S} \xleftarrow{\cup} \{(M, \sigma)\}$
$(M^*, \sigma^*) \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Sign}}(\mathsf{par}, \mathsf{vk})$	Return σ
If $(M^*, \sigma^*) \notin \mathsf{S} \wedge \mathsf{SS.Ver}(\mathsf{par}, \mathsf{vk}, M^*, \sigma^*)$ then	
Return `true`	
Return `false`	

Fig. 1. Strong unforgeability.

2 Preliminaries

NOTATION. We will use the shorthand $[n] = \{1, \ldots, n\}$. We will denote formal variables in polynomials with sans-serif letters $\mathsf{X}, \mathsf{Y}, \ldots$, and for any modulus p, we let $\mathbb{Z}_p[\mathsf{X}]$ be the ring of formal polynomials $a(\mathsf{X}) = \sum_{i=0}^{d} a_i \mathsf{X}^i$ with coefficients in $\mathbb{Z}_p$. As usual, d is the *degree* of $a(\mathsf{X})$.

Throughout the paper, we adopt as far as possible the concrete security and efficiency approach, and avoid qualitative statements. We refer to "efficient" informally to stress that an algorithm is meant to be as efficient as possible, but make theorems precise by giving explicit reductions in their proofs.

GROUPS AND PAIRINGS. We work with prime-order groups. For such a group $\mathbb{G}$, we denote by $1_{\mathbb{G}}$ the identity element, and let $\mathbb{G}^* = \mathbb{G} \backslash \{1_{\mathbb{G}}\}$ be the set of $p-1$ generators. We use multiplicative notation, and generally denote group elements with upper case letters, scalars with lower case ones, with the exception of generators. For a generator $g \in \mathbb{G}^*$ and a group element $X \in \mathbb{G}$, we also let $\mathrm{DL}_g(X)$ be the discrete logarithm $x \in \mathbb{Z}_p$ of X to the base g, i.e., $g^x = X$.

For prime-order groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, a *bilinear map* is an efficiently computable function $\mathsf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ which satisfies both (1) *bi-linearity*, i.e., $\mathsf{e}(A^x, B^y) = \mathsf{e}(A, B)^{xy}$ for all $A \in \mathbb{G}_1$, $B \in \mathbb{G}_2$, and $x, y \in \mathbb{Z}_p$, and (2) *non-triviality*, i.e., $\mathsf{e}(g_1, g_2) \in \mathbb{G}_T^*$ for all generators $g_1 \in \mathbb{G}_1^*$ and $g_2 \in \mathbb{G}_2^*$. We normally consider a *group parameters generation algorithm* GGen such that $\mathsf{GGen}(1^\lambda)$ outputs a description $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e})$ such that $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of order p, $\mathsf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map, and p is a λ-bit prime.

Our treatment is compatible with type-3 pairings (cf. e.g. [GPS06]) like BLS curves [BLS03] which allow for some of the most efficient implementations, e.g., using BLS12-381 [Bow17]. The most relevant property is that $\mathbb{G}_1 \neq \mathbb{G}_2$ and no efficiently computable homomorphism $\mathbb{G}_2 \to \mathbb{G}_1$ exists.

SIGNATURE SCHEMES. A *signature scheme* SS consists of a setup algorithm $\mathsf{SS.Setup}$, a *key generation algorithm* $\mathsf{SS.KG}$, a (possibly randomized) *signing algorithm* $\mathsf{SS.Sign}$, and a deterministic *verification algorithm* $\mathsf{SS.Ver}$, satisfying the usual syntax and correctness requirement. In particular, $\mathsf{SS.Setup}$ outputs parameters par, upon which all algorithms depend. We also let the message space $\mathsf{SS.M} = \mathsf{SS.M}(\mathrm{par})$ depend on par. (We implicitly assume that the signing algorithm returns an error symbol $\bot$, which is not a valid signature, if the message is not in the message space.) We target *strong unforgeability*, which is defined by Game $\mathsf{SUF}^{\mathcal{A}}_{\mathsf{SS}}(\lambda)$ in Figure 1. The corresponding advantage metric is

$$\mathsf{Adv}^{\mathsf{suf}}_{\mathsf{SS}}(\mathcal{A}, \lambda) = \Pr\left[\mathsf{SUF}^{\mathcal{A}}_{\mathsf{SS}}(\lambda)\right] .$$

THE SECURITY ASSUMPTIONS. We will use the following variant of the *q-Strong Diffie-Hellman* (q-SDH) assumption, as defined by Boneh and Boyen [BB08] in a format meant to support type-3

Game q-$\mathsf{SDH}^{\mathcal{A}}_{\mathsf{GGen}}(\lambda)$:	Game q-$\mathsf{DL}^{\mathcal{A}}_{\mathsf{GGen}}(\lambda)$:
$\mathrm{par} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}) \leftarrow \mathsf{GGen}(1^\lambda)$	$\mathrm{par} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}) \leftarrow \mathsf{GGen}(1^\lambda)$
$g_1 \leftarrow\!\!\$\ \mathbb{G}_1^*,\ g_2 \leftarrow\!\!\$\ \mathbb{G}_2^*$	$g_1 \leftarrow\!\!\$\ \mathbb{G}_1^*,\ g_2 \leftarrow\!\!\$\ \mathbb{G}_2^*$
$x \leftarrow\!\!\$\ \mathbb{Z}_p$	$x \leftarrow\!\!\$\ \mathbb{Z}_p$
$(c, Z) \leftarrow\!\!\$\ \mathcal{A}(\mathrm{par}, g_1, (g_1^{x^i})_{i\in[q]}, g_2, g_2^x)$	$x' \leftarrow\!\!\$\ \mathcal{A}(\mathrm{par}, g_1, (g_1^{x^i})_{i\in[q]}, g_2, g_2^x)$
Return $Z = g_1^{1/(x+c)}$	Return $x' = x$

Fig. 2. Assumptions. The assumptions could also be defined with respect to fixed generators, but this would invalidate some of our security proofs.

pairings. It is formalized by Game q-$\mathsf{SDH}^{\mathcal{A}}_{\mathsf{GGen}}(\lambda)$ on the left of Figure 2. We also consider the related *q-Discrete Logarithm* (q-DL) assumption, as formulated on the right of Figure 2 by Game q-$\mathsf{DL}^{\mathcal{A}}_{\mathsf{GGen}}(\lambda)$, which only differs in the winning condition. We associate with these games the corresponding advantage metrics

$$\mathsf{Adv}^{q\text{-}\mathsf{sdh}}_{\mathsf{GGen}}(\mathcal{A}, \lambda) = \mathsf{Pr}\left[q\text{-}\mathsf{SDH}^{\mathcal{A}}_{\mathsf{GGen}}(\lambda)\right] \ , \ \ \mathsf{Adv}^{q\text{-}\mathsf{dl}}_{\mathsf{GGen}}(\mathcal{A}, \lambda) = \mathsf{Pr}\left[q\text{-}\mathsf{DL}^{\mathcal{A}}_{\mathsf{GGen}}(\lambda)\right] \ . \tag{1}$$

We note that the q-SDH assumption implies the q-DL assumption for any q, as finding x implies finding $g_1^{1/(x+c)}$ for any c. The converse is not known to be true *in general*, but it is true for algebraic adversaries [BFL20]. Notation-wise, we drop q whenever it equals one, and refer to the resulting assumption as the *Discrete Logarithm* (DL) assumption.

Remark 1. Our security proofs will repeatedly rely on the observation (due to Boneh and Boyen [BB08]) that, given $g_1, g_1^x, g_1^{x^2}, \ldots, g_1^{x^q}$, computing $A = g_1^{p(x)/(x-e)}$, for any known non-zero polynomial $p(\mathsf{X}) \in \mathbb{Z}_p[X]$ with degree at most q such that $p(e) \neq 0$, leads to a break q-SDH. This is because, by the polynomial remainder theorem, we can write $p(\mathsf{X}) = d(\mathsf{X})(X - e) + r$, where the remainder $r = p(e) \in \mathbb{Z}_p$ is a non-zero integer mod p, whereas $d(\mathsf{X})$ has degree at most $q - 1$. Therefore, $A = g_1^{d(x)+r/(x-e)}$, and also,

$$(A g_1^{-d(x)})^{1/r} = g_1^{\frac{1}{x-e}}$$

can be efficiently computed, and $(-e, g_1^{1/(x-e)})$ is a q-SDH solution.

3 New Proof for (Short) BBS Signatures

3.1 Description and implementation details

Figure 3 describes a version of BBS with shorter signatures than BBS+ [ASM06]. We refer formally to this scheme as $\mathsf{BBS} = \mathsf{BBS}[\mathsf{GGen}, \mathcal{D}_e, \ell]$, where GGen is a group parameter generator, $\mathcal{D}_e$ is a distribution over $\mathbb{Z}_p$, and $\ell \geqslant 1$ a parameter. We omit $\mathcal{D}_e$ whenever it is understood to be the uniform distribution over $\mathbb{Z}_p$, and ℓ whenever it is clear from the context. Here, the message space is $\mathsf{BBS.M} = \mathbb{Z}_p^\ell$, and it depends on the parameters in that the modulus p is determined by GGen via $\mathsf{BBS.Setup}$. There is an unlikely event that the inversion to compute $1/(x + e)$ during signature issuance fails because $x + e = 0$—for ease of syntax, we use the convention that $1/0 = 0$. The BBS^+ scheme is a special case where each signed message $\boldsymbol{m}$ is such that its first component, $\boldsymbol{m}[1]$, is randomly chosen. (And, therefore, needs to be made part of the signature.)

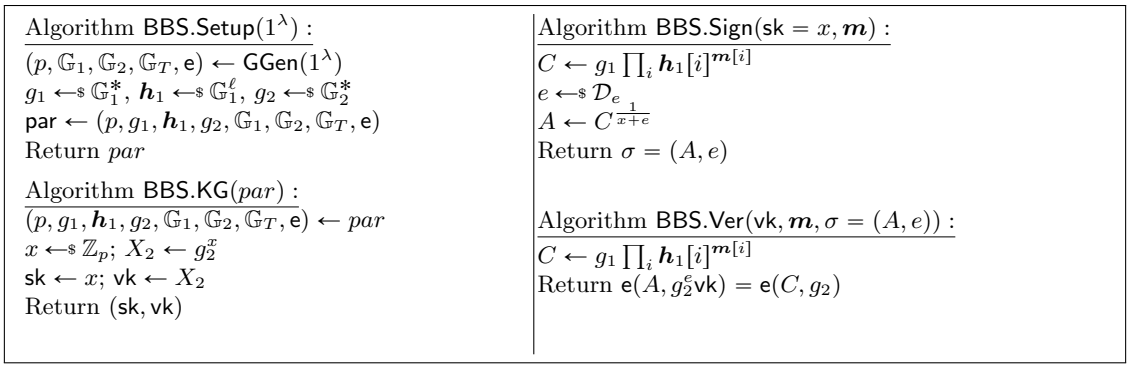

Fig. 3. BBS signature. The scheme is parameterized by GGen, $\mathcal{D}_e$, and the message length $\ell = \ell(\lambda) \geqslant 1$. Here, group operations are in the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ determined by the parameters.

Modeling choices. Our modeling is similar to that of [BBS04, ASM06, CDL16], in that in particular we fix the message length via ℓ. One can easily accommodate unbounded-length messages, as in practice, the generators in $\boldsymbol{h}_1$ do not need to be fixed beforehand, and $\boldsymbol{h}_1[i]$ can be the output of a hash function (modeled as a random oracle) on some input that depends on i. This allows us to also sign messages in $\mathbb{Z}_p^+$, given a suitable encoding. (The RFC draft [LKWL22] suggests hashing a length-dependent set of parameters into the first message block, although more efficient encodings certainly exist.)

We also model BBS as randomized, as this feature may be useful in some contexts, but we can de-randomize the scheme by applying a PRF to $\boldsymbol{m}$, or (more efficiently) to C, to derive e.

3.2 Security Analysis

We show security of BBS in the standard model, under the q-SDH assumption, for the setting where $\mathcal{D}_e$ is the uniform distribution over $\mathbb{Z}_p$. Here, q is the number of signing queries issued by the signer. Note that this theorem *also* implies security of BBS+, as it corresponds to a special case of BBS where the first block of each signed message is randomly chosen, and included in the signature.

Theorem 1 (Security of BBS). *Let* GGen *be a group parameter generator, producing groups of order* $p = p(\lambda)$*. For every SUF adversary* $\mathcal{A}$ *issuing at most* $q = q(\lambda)$ *signing queries, there exist adversaries* $\mathcal{B}_1$*,* $\mathcal{B}_2$*, and* $\mathcal{B}_3$ *such that*

$$\mathsf{Adv}^{\mathsf{suf}}_{\mathsf{BBS}[\mathsf{GGen}]}(\mathcal{A}, \lambda) \leqslant q \cdot \mathsf{Adv}^{q\text{-}\mathsf{sdh}}_{\mathsf{GGen}}(\mathcal{B}_1, \lambda) + \mathsf{Adv}^{\mathsf{dl}}_{\mathsf{GGen}}(\mathcal{B}_2, \lambda) + \mathsf{Adv}^{q\text{-}\mathsf{sdh}}_{\mathsf{GGen}}(\mathcal{B}_3, \lambda) + \frac{q^2}{2p} + \frac{q+2}{p} \ .$$

The adversaries $\mathcal{B}_1$*,* $\mathcal{B}_2$ *and* $\mathcal{B}_3$ *are given explicitly in the proof, and run in time roughly comparable to that of* $\mathcal{A}$*.*

The proof of the theorem is given in Section 3.3 below. The concrete bound is essentially the same as prior analyses of BBS+ [ASM06, CDL16], and we incur a factor q loss in the reduction. Below, in Theorem 2, we give a tight reduction to q-DL in the algebraic group model, which suggests this loss may be artificial.

Discussion of concrete parameters. Even assuming the tight bound as the correct one, the reliance on q-SDH raises the question of the extent to which parameters should accommodate for Cheon's attack [Che06] on q-SDH/q-DL, which achieves complexity (roughly) $O(\sqrt{p/q})$ for certain choices of q. The RFC [LKWL22] suggests the use of BLS12-381 [Bow17], which gives (roughly) a 256-bit group order. We could accommodate for roughly $q = 2^{36}$, for example, while still having 110-bit security. (This type of reasoning was for example adopted to justify BLS12-381 in zkSNARKs [zca].) But even then, we observe that the only way we know to *actually* break BBS via Cheon's attack is the reduction by Jao and Yoshida [JY09], which requires all signatures to be on the same message, with different e-values.[1] Not only this situation is unlikely to arise, but it *does not* occur if we de-randomize BBS, which is the choice the RFC also adopted for BBS+. It is an excellent question to see whether a similar attack exists even for de-randomized BBS.

3.3 Proof of Theorem 1

Let us consider an interaction of the adversary $\mathcal{A}$ in the SUF game, where the adversary finally outputs a forgery $(\boldsymbol{m}^*, \sigma^*)$, where $\sigma^* = (A^*, e^*)$. We define three events, depending on the specific format of the forgery:

- Forge_1: This is the event where σ^* is a valid forgery, and a prior Sign query has output a signature $\sigma_i = (A_i, e_i)$ for $A_i \neq A^*$, $e_i = e_i^*$, and some message $\boldsymbol{m}_i \neq \boldsymbol{m}^*$.
- Forge_2: This is the event where σ^* is a valid forgery, and a prior Sign query output the *same* signature $\sigma_i = \sigma^*$ for a message $\boldsymbol{m}_i \neq \boldsymbol{m}^*$, or the forgery A^* equals $1_{\mathbb{G}_1}$.
- Forge_3: This is the event where σ^* is a valid forgery and completely fresh, i.e., neither of Forge_1 or Forge_2 occurs.

As the union of these three events equal the event that $(\boldsymbol{m}^*, \sigma^*)$ is a valid forgery, the union bound yields

$$\mathsf{Adv}^{\mathsf{suf}}_{\mathsf{BBS}}(\mathcal{A}, \lambda) \leqslant \mathsf{Pr}\left[\mathsf{Forge}_1\right] + \mathsf{Pr}\left[\mathsf{Forge}_2\right] + \mathsf{Pr}\left[\mathsf{Forge}_3\right] \ .$$

We will proceed in upper bounding these three probabilities via separate reductions. The hardest case is the analysis of Forge_1, and this is where out proof differs from prior work. The analyses of Forge_2 and Forge_3 are essentially the same as in the original analysis of BBS+. The theorem statement then follows by combining Lemmas 1, 2, and 3, which we state next. The proofs are given below.

Lemma 1 (Analysis of Forge_1). *There exists a q-sdh adversary $\mathcal{B}_1$ such that*

$$\mathsf{Pr}\left[\mathsf{Forge}_1\right] \leqslant q \cdot \mathsf{Adv}^{q\text{-}\mathsf{sdh}}_{\mathsf{GGen}}(\mathcal{B}_1, \lambda) + \frac{q^2}{2p} + \frac{1}{p} \ .$$

Lemma 2 (Analysis of Forge_2). *There exists a dl adversary $\mathcal{B}_2$ such that*

$$\mathsf{Pr}\left[\mathsf{Forge}_2\right] \leqslant \mathsf{Adv}^{\mathsf{dl}}_{\mathsf{GGen}}(\mathcal{B}_2, \lambda) + \frac{1}{p} \ .$$

Lemma 3 (Analysis of Forge_3). *There exists a q-sdh adversary $\mathcal{B}_3$ such that*

$$\mathsf{Pr}\left[\mathsf{Forge}_3\right] \leqslant \mathsf{Adv}^{q\text{-}\mathsf{sdh}}_{\mathsf{GGen}}(\mathcal{B}_3, \lambda) + \frac{q}{p} \ .$$

[1] Roughly, their attack considers the setting where $g^{\frac{1}{x+e_i}}$ is obtained for multiple e_i's.

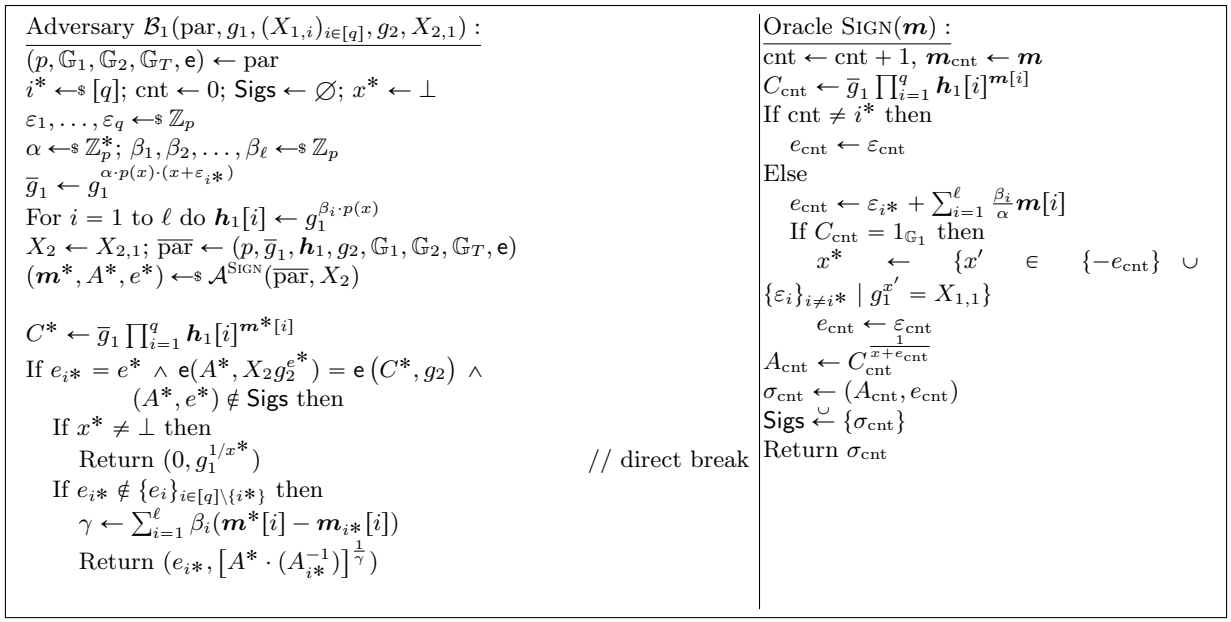

Fig. 4. Adversary $\mathcal{B}_1$ in the proof of Lemma 1. Recall that once $\varepsilon_1, \ldots, \varepsilon_q$ are fixed and understood from the context, we use the shorthand $p(\mathsf{X}) = \prod_{i\in[q]\setminus\{i^*\}}(X + \varepsilon_i)$ for convenience. In the pseudo-code, we omit the explicit computations of $\overline{g}$, $\boldsymbol{h}_1$, and A_{cnt} from C_{cnt}, which are detailed in the text.

3.4 Proof of Lemma 1

We give an overview of the adversary $\mathcal{B}_1$ that underlies the reduction to q-SDH for Forge_1. The formal pseudocode description is in Figure 4, although we omit there some lengthier and tedious descriptions of how to compute certain elements, and give them here in the text instead. Recall that q is a bound on the number of generated signatures, i.e., the number of queries to Sign issued by the adversary $\mathcal{A}$. We assume here that exactly q queries are made, without loss of generality.

Given the q-SDH setup $g_1, X_{1,1} = g_1^x, \ldots, X_{1,q} = g_1^{x^q}, g_2, X_2 = g_2^x$, the adversary $\mathcal{B}_1$ first generates a suitable setup to run $\mathcal{A}$. In particular, it picks random values $\varepsilon_1, \ldots, \varepsilon_q \leftarrow\$ \mathbb{Z}_p$, as well as randomizers $\alpha \leftarrow\$ \mathbb{Z}_q^*$ and $\beta_1, \ldots, \beta_q \leftarrow\$ \mathbb{Z}_q$. Then, the generators $\overline{g}_1 \in \mathbb{G}_1^*$ and $\boldsymbol{h}_1 \in \mathbb{G}_1^\ell$ are set to

$$\overline{g}_1 = g_1^{\alpha\cdot p(x)\cdot(x+\varepsilon_{i^*})}, \quad \boldsymbol{h}_1[i] \leftarrow g_1^{\beta_i \cdot p(x)} \text{ for all } i \in [\ell],$$

where $i^* \leftarrow\$ [q]$ and

$$p(\mathsf{X}) = \prod_{i\in[q]\setminus\{i^*\}} (\mathsf{X} + \varepsilon_i) .$$

It is not hard to see that $\overline{g}_1$ and $\boldsymbol{h}_1$ can be computed efficiently from part of the q-SDH setup $g_1, X_{1,1}, \ldots, X_{1,q}$. Moreover, at least informally, it should be clear that as long as $x \notin \{-\varepsilon_1, \ldots, -\varepsilon_q\}$, the distributions of $\overline{g}_1$ and $\boldsymbol{h}_1$ are correct, i.e., they are uniform in $\mathbb{G}_1^*$ and $\mathbb{G}_1^\ell$, respectively, since $g_1 \in \mathbb{G}_1^*$. (The formal argument about the correctness of distributions is given below, and this is only meant to serve as some intuition.) We stress that our simulation will not be correct if $x \in \{-\varepsilon_1, \ldots, -\varepsilon_q\}$, so it is easiest to assume that this is not the case to understand the rest of the reduction.

HANDLING SIGNING QUERIES. The oracle SIGN then simulates the correct signing oracle, keeping a query counter cnt. Whenever cnt $\neq i^*$, it is not hard to see that SIGN can easily answer the query using $e_{\text{cnt}} = \varepsilon_{\text{cnt}}$. Indeed, if $x + \varepsilon_{\text{cnt}} \neq 0$, on input $\boldsymbol{m}$, the simulate SIGN can compute

$$A_{\text{cnt}} = \left(\overline{g}_1 \prod_{i=1}^{\ell} \boldsymbol{h}_1[i]^{\boldsymbol{m}[i]} \right)^{\frac{1}{x+\varepsilon_{\text{cnt}}}} = g_1^{p_i(x)\left[\alpha(x+\varepsilon_{i^*})+\sum_{i=1}^{\ell}\beta_i \boldsymbol{m}[i]\right]} ,$$

where $p_i(x) = \prod_{i \notin \{\text{cnt}, i^*\}}(x + \varepsilon_i)$. It is easy to detect $x + \varepsilon_{\text{cnt}} = 0$, and in that case, $A_{\text{cnt}} = 1_{\mathbb{G}_1}$ by definition.

Crucially, when cnt $= i^*$ the adversary $\mathcal{B}_1$ answers the signing query differently. We observe first that, with $C_{\text{cnt}} \leftarrow \overline{g}_1 \prod_{i=1}^{q} \boldsymbol{h}_1[i]^{\boldsymbol{m}[i]}$,

$$\begin{aligned} \text{DL}_{g_1}(C_{\text{cnt}}) &= \alpha \cdot p(x) \cdot (x + \varepsilon_{i^*}) + \sum_{i=1}^{\ell} \beta_i \boldsymbol{m}[i] p(x) \\ &= \alpha \cdot p(x) \left(x + \varepsilon_{i^*} + \sum_{i=1}^{\ell} \frac{\beta_i}{\alpha} \boldsymbol{m}[i] \right) . \end{aligned} \tag{2}$$

Here, there are two cases. Either $C_{\text{cnt}} = 1_{\mathbb{G}_1}$, and then we return $A_{\text{cnt}} = 1_{\mathbb{G}_1}$, along with a $e_{\text{cnt}} = \varepsilon_{\text{cnt}}$. Alternatively, and more interestingly, if $C_{\text{cnt}} \neq 1_{\mathbb{G}_1}$, we set $e_{i^*} = \varepsilon_{i^*} + \sum_{i=1}^{\ell} \frac{\beta_i}{\alpha} \boldsymbol{m}[i]$, and

$$A_{i^*} = C_{i^*}^{\frac{1}{x+e_{i^*}}} = g_1^{\alpha \cdot p(x)} ,$$

which can be efficiently computed. The bulk of our analysis below will show that if $C_{i^*} \neq 1_{\mathbb{G}_1}$, then we indeed generate a random e_{i^*} in this way.

Note that by equation (2) if $C_{i^*} = 1_{\mathbb{G}_1}$, then $x = -\varepsilon_{i^*} - \sum_{i=1}^{\ell} \frac{\beta_i}{\alpha} \boldsymbol{m}[i]$ or $x \in \{\varepsilon_i\}_{i \neq i^*}$, and hence we can directly break of q-SDH. (The variable x^* here stores the recovered discrete logarithm.) To simplify the analysis below, in this case, it is convenient for the reduction $\mathcal{B}_1$ to defer breaking q-SDH to end, and return the signature $(1_{\mathbb{G}_1}, \varepsilon_{i^*})$ instead.

EXTRACTING A SOLUTION. Assume now that $\mathcal{A}$ outputs a valid forgery $\boldsymbol{m}^*, \sigma^*$, where $\sigma^* = (A^*, e^*)$, $e^* = e_{i^*}$, and $A^* \neq A_{i^*}$. Further, assume that $C_{i^*} \neq 1_{\mathbb{G}_1}$, which implies that $x + e_{i^*} \neq 0$ and $p(x) \neq 0$. (If this was not true, as highlighted above, we would have extracted x already.) Then,

$$\begin{aligned} \text{DL}_{g_1}(A^*) &= \alpha p(x) \frac{x + \varepsilon_{i^*} + \sum_{i=1}^{\ell} \frac{\beta_i}{\alpha} \boldsymbol{m}^*[i]}{x + \varepsilon_{i^*} + \sum_{i=1}^{\ell} \frac{\beta_i}{\alpha} \boldsymbol{m}[i]} \\ &= \alpha \cdot p(x) + p(x) \frac{\sum_{i=1}^{\ell} \beta_i (\boldsymbol{m}^*[i] - \boldsymbol{m}[i])}{x + e_{i^*}} . \end{aligned}$$

Further, because $A_{i^*} \neq A^*$ but $e^* = e_{i^*}$, we also have $\gamma = \sum_{i=1}^{\ell} \beta_i (\boldsymbol{m}^*[i] - \boldsymbol{m}[i]) \neq 0$, and then

$$\left[A^* \cdot (A_{i^*}^{-1}) \right]^{\frac{1}{\gamma}} = g_1^{\frac{p(x)}{x+e_{i^*}}} .$$

If $e_{i^*} \notin \{e_i\}_{i \in [q] \setminus \{i^*\}}$, $\mathsf{X} + e_{i^*}$ does not divide $p(\mathsf{X})$. We can then compute $g_1^{\frac{1}{x+e_{i^*}}}$ using Remark 1 and break q-SDH.

FORMAL ANALYSIS. We now proceed with a formal analysis to show that the probability guarantees for $\mathcal{B}_1$ as stated in the lemma indeed hold. To this end, we use $\mathsf{Pr}_0[\cdot]$ to denote probabilities in the experiment $\mathsf{SUF}^{\mathcal{A}}_{\mathsf{BBS}}(\lambda)$, where $\mathcal{A}$ plays the SUF game against BBS. Similarly, we use $\mathsf{Pr}_1[\cdot]$ to denote probabilities in the simulated experiment where $\mathcal{A}$ is run within $\mathcal{B}_1$ in Game q-$\mathsf{SDH}^{\mathcal{B}_1}_{\mathsf{GGen}}(\lambda)$.

In both experiments, we can define the event Forge_1, as it only depends on the output of the adversary and the property of this output relative to its earlier signing query. Moreover, let $\mathsf{Forge}_1^{(i)}$ for $i \in [q]$ be the event that Forge_1 happens and the i-th query is the first signing query that satisfies the condition for Forge_1 to happen. Let GoodE be the event that all e_i's are distinct. Note that $\mathcal{B}_1$ is guaranteed to output a q-SDH solution if $\mathsf{Forge}_1^{(i)}$ happens and $i = i^*$ and, moreover, GoodE also occurs. Also, note that the probability that GoodE and $\mathsf{Forge}_1^{(i)}$ occurs is independent of whether $i = i^*$ or not, and therefore

$$\begin{aligned}\mathsf{Adv}^{q\text{-}\mathsf{sdh}}_{\mathsf{GGen}}(\mathcal{B}_1,\lambda) &\geqslant \sum_{i=1}^{q} \mathsf{Pr}_1\left[\mathsf{GoodE} \wedge \mathsf{Forge}_1^{(i)}\right] \cdot \mathsf{Pr}_1\left[i^* = i\right] \\ &= \frac{1}{q} \cdot \sum_{i=1}^{q} \mathsf{Pr}_1\left[\mathsf{GoodE} \wedge \mathsf{Forge}_1^{(i)}\right] = \frac{1}{q} \cdot \mathsf{Pr}_1\left[\mathsf{GoodE} \wedge \mathsf{Forge}_1\right] \ .\end{aligned}$$

We rely on the following central lemma, which in particular shows that the simulation of $\mathcal{A}$'s execution within $\mathcal{B}$ is nearly correct. While the intuition has been given above, the formal proof is rather subtle and we rely on the H-coefficient method [Pat08, CS14] to prove the following.

Lemma 4. $\mathsf{Pr}_0\left[\mathsf{GoodE} \wedge \mathsf{Forge}_1\right] - \mathsf{Pr}_1\left[\mathsf{GoodE} \wedge \mathsf{Forge}_1\right] \leqslant \frac{q}{p}$.

Before turning to a proof of the lemma in Section 3.5 below, we observe that plugging the inequality of the lemma into the above yields

$$\begin{aligned}\mathsf{Adv}^{q\text{-}\mathsf{sdh}}_{\mathsf{GGen}}(\mathcal{B}_1,\lambda) &\geqslant \frac{1}{q} \cdot \mathsf{Pr}_0\left[\mathsf{GoodE} \wedge \mathsf{Forge}_1\right] - \frac{1}{p} \\ &\geqslant \frac{1}{q}\left(\mathsf{Pr}_0\left[\mathsf{Forge}_1\right] - \mathsf{Pr}_0\left[\overline{\mathsf{GoodE}}\right]\right) - \frac{1}{p} \ .\end{aligned}$$

On the other hand, $\mathsf{Pr}_0\left[\overline{\mathsf{GoodE}}\right] \leqslant \binom{q}{2}\frac{1}{p} \leqslant \frac{q^2}{2p}$, and thus we obtain the bound in Lemma 1 by re-arranging terms.

3.5 Proof of Lemma 4

We assume $\mathcal{A}$ to be deterministic without loss of generality. We describe the transcripts of the interaction of $\mathcal{A}$ in the SUF and within $\mathcal{B}_1$ as part of the q-SDH experiment, respectively, via the following two random variables

$$\begin{aligned}T_0 &= (g_1, g_2, \boldsymbol{h}_1, x, i^*, (\boldsymbol{m}_1, e_1), \ldots, (\boldsymbol{m}_q, e_q)) \ , \\ T_1 &= (\overline{g}_1, g_2, \boldsymbol{h}_1, x, i^*, (\boldsymbol{m}_1, e_1), \ldots, (\boldsymbol{m}_q, e_q))\end{aligned}$$

where in T_0, i^* is sampled uniformly from $[q]$, independently of everything else. We do not include X_2, as $X_2 = g_2^x$ in both experiments. Moreover, in both experiments, the first component $A_1, A_2, \ldots$ of the the answer to each signature query is removed, as it is also a deterministic function of the rest

of the transcript. Similarly, the final forgery (A^*, e^*) is also a function of T_0/T_1. For this reason, we note that that the event $\mathsf{GoodE} \wedge \mathsf{Forge}_1$ is deterministically determined from T_0 and T_1, in their respective experiments, i.e., there exists a Boolean function ϕ such that $\phi(T_b) = 1$ if and only if the event happens in the corresponding experiment. Therefore,

$$\mathsf{Pr}_0\left[\mathsf{GoodE} \wedge \mathsf{Forge}_1\right] - \mathsf{Pr}_1\left[\mathsf{GoodE} \wedge \mathsf{Forge}_1\right] \leqslant \mathsf{SD}(T_0, T_1) \ ,$$

where $\mathsf{SD}(T_0, T_1) = \frac{1}{2}\sum_\tau \left|\mathsf{Pr}\left[T_0 = \tau\right] - \mathsf{Pr}\left[T_1 = \tau\right]\right|$ is the total variation distance, which we upper bound by a special case of Patarin's H-coefficient method [Pat08], which we introduce on the way. (We use the formalism from [HT16] here.)

Interpolation probabilities. Concretely, for any potential value τ of T_b for $b \in \{0, 1\}$, which we denote as

$$\tau = (\underline{g}_1, \underline{g}_2, \underline{\boldsymbol{h}}_1, \underline{x}, \underline{i}^*, (\underline{\boldsymbol{m}}_1, \underline{e}_1), \ldots, (\underline{\boldsymbol{m}}_q, \underline{e}_q)) \ ,$$

we let $\mathsf{p}_0(\tau)$ and $\mathsf{p}_1(\tau)$ be its interpolation probabilities, i.e., the probabilities that we pick randomness in the respective experiment that would lead to transcript τ if queries $\underline{\boldsymbol{m}}_1, \ldots, \underline{\boldsymbol{m}}_q$ are fixed ahead of time. (These probabilities are independent of $\mathcal{A}$, and only depend on τ and the randomness of the experiment.) We want to isolate the following type of good transcript.

Definition 1 (Good transcript). *We call* τ good *if* $\underline{x} \notin \{-\underline{e}_1, \ldots, -\underline{e}_q\}$. *Otherwise,* τ *is* bad.

We are then going to prove that for all good transcripts τ, $\mathsf{p}_0(\tau) = \mathsf{p}_1(\tau)$. This is enough to conclude the proof, as it implies that

$$\mathsf{SD}(T_0, T_1) \leqslant \mathsf{Pr}\left[T_0 \text{ is bad}\right] = \mathsf{Pr}_0\left[x \in \{-e_1, \ldots, -e_q\}\right] \leqslant \frac{q}{p} \ .$$

To compute $\mathsf{p}_1(\tau)$ for a good transcript τ, we assume that the generator g_1 given to $\mathcal{B}_1$ is fixed. (Of course, it is actually sampled randomly from $\mathbb{G}_1^*$ as part of the q-SDH instance, but the interpolation probability is the same conditioned on any particular choice, and thus we fix it.) The randomness then consists of $i^* \leftarrow\!\!\$\, [q]$, $x \leftarrow\!\!\$\, \mathbb{Z}_p$, $\alpha \leftarrow\!\!\$\, \mathbb{Z}_p^*$, $\varepsilon_1, \ldots, \varepsilon_q \leftarrow\!\!\$\, \mathbb{Z}_p$, and $g_2 \leftarrow\!\!\$\, \mathbb{G}_2^*$. To generate the transcript τ, we need in particular

$$i^* = \underline{i}^*, \ g_2 = \underline{g}_2, \ x = \underline{x}, \ (\varepsilon_i)_{i \in [q] \setminus \{\underline{i}^*\}} = (\underline{e}_i)_{i \in [q] \setminus \{\underline{i}^*\}}$$

and as these values are chosen independently, this is true with probability

$$\frac{1}{q} \cdot \frac{1}{p-1} \cdot \frac{1}{p} \cdot \frac{1}{p^{q-1}} = \frac{1}{q(p-1)p^q}$$

over the choice of $i^*, g_2, x, \{\varepsilon_i\}_{i \in [q] \setminus \{\underline{i}^*\}}$. Let us assume this initial part of the transcript is consistent. We also need $\beta_1, \ldots, \beta_\ell$ to ensure $\boldsymbol{h}_1 = \underline{\boldsymbol{h}}_1$, which, conditioned on $x = \overline{x}$, is equivalent to the fact that

$$p(\underline{x}) \cdot \beta_i = \mathrm{DL}_{g_1}(\underline{\boldsymbol{h}}_1[i]) \quad \text{for all } i \in [\ell].$$

Because τ is good, $p(\underline{x}) \neq 0$, and therefore the ℓ equalities hold with probability $1/p^\ell$ over the choice of $\beta_1, \ldots, \beta_\ell$.

Finally, conditioned on $i^*, g_2, x, \{\varepsilon_i\}_{i \in [q] \setminus \{\underline{i}^*\}}, \{\beta\}_{i \in [\ell]}$ satisfying all above constraints, we need $\varepsilon_{\underline{i}^*}$ and α to ensure that

$$\overline{g}_1 = g_1^{\alpha p(\underline{x})(\underline{x} + \varepsilon_{\underline{i}^*})} = \underline{g}_1 \ , \quad e_{\underline{i}^*} = \underline{e}_{\underline{i}^*} \ .$$

There are two cases here, depending on $\underline{\boldsymbol{m}} = \underline{\boldsymbol{m}}_{\underline{i}^*}$ and the associated value

$$\underline{C} = \underline{g}_1 \prod_{i=1}^{\ell} \underline{\boldsymbol{h}}_1[i]^{\underline{\boldsymbol{m}}[i]} \ .$$

Case 1: $\underline{C} = 1_{\mathbb{G}_1}$. Then, in this case, $\mathcal{B}_1$ ensures $e_{\underline{i}^*} = \varepsilon_{\underline{i}^*}$, and this value, which is uniform, equals $\underline{e}_{\underline{i}^*}$ with probability $1/p$. Conditioned on this,

$$p(\underline{x})(\underline{x} + \varepsilon_{\underline{i}^*}) = p(\underline{x})(\underline{x} + \underline{e}_{\underline{i}^*}) \neq 0$$

because τ is good. Thus $\underline{g}_1 = g_1^{\alpha p(\underline{x})(\underline{x}+\varepsilon_{\underline{i}^*})}$ holds with probability $1/(p-1)$ over the choice of α from $\mathbb{Z}_p^*$.

Case 2: $\underline{C} \neq 1_{\mathbb{G}_1}$. For convenience, we write $\underline{e} = \underline{e}_{\underline{i}^*}$, $\varepsilon = \varepsilon_{\underline{i}^*}$, $\underline{a} = \mathrm{DL}_{g_1}(\underline{g}_1)$, and $\underline{y} = \sum_i \beta_i \underline{\boldsymbol{m}}[i]$. Here, $\alpha \leftarrow\!\!\$\ \mathbb{Z}_p^*$ and $\varepsilon \leftarrow\!\!\$\ \mathbb{Z}_p$ need to satisfy

$$\alpha p(\underline{x})(\underline{x} + \varepsilon) = \underline{a}$$
$$\varepsilon + \frac{1}{\alpha}\underline{y} = \underline{e}$$

The second equation directly implies that

$$\varepsilon = \underline{e} - \frac{1}{\alpha} \cdot \underline{y} \ . \tag{3}$$

Substituting this into the first equation yields

$$\alpha \cdot p(\underline{x}) \left(\underline{x} + \underline{e} - \frac{1}{\alpha} \cdot \underline{y} \right) = p(\underline{x})(\alpha(\underline{x} + \underline{e}) - \underline{y}) = \underline{a} \ .$$

Re-arranging terms we get

$$\alpha = \frac{\underline{a}/p(\underline{x}) + \underline{y}}{\underline{x} + \underline{e}} \ . \tag{4}$$

This is indeed a value in $\mathbb{Z}_p^*$ for two reasons. First off, $\underline{x}+\underline{e} \neq 0$ as τ is good. Second, $\underline{a}/p(\underline{x})+\underline{y} \neq 0$. Indeed, if instead $\underline{a}/p(\underline{x}) + \underline{y} = 0$ were true, we would have

$$\underline{C} = \underline{g}_1 \prod_{i=1}^{\ell} \underline{\boldsymbol{h}}_1[i]^{\underline{\boldsymbol{m}}[i]} = g_1^{\underline{a}} \prod_{i=1}^{\ell} g_1^{p(\underline{x}) \cdot \beta_i \cdot \underline{\boldsymbol{m}}[i]} = g_1^{\underline{a} + p(\underline{x})\underline{y}} = g_1^0 = 1_{\mathbb{G}_1} \ ,$$

a contradiction with the fact that we are in Case 2. Therefore, the probability over the choice of α, ε that (3) and (4) are both satisfied is $\frac{1}{p(p-1)}$.

Therefore, in summary, we have

$$\mathsf{p}_1(\tau) = \frac{1}{q(p-1)p^q} \cdot \frac{1}{p^\ell} \cdot \frac{1}{p(p-1)} = \frac{1}{q(p-1)^2 p^{q+\ell+1}} \ .$$

It is not hard to observe that we also have

$$\mathsf{p}_0(\tau) = \frac{1}{p^{q+\ell+1}(p-1)^2 q} \ ,$$

because g_1, g_2 are uniform over $\mathbb{G}_1^*$, $\overline{h}_1$ is uniform over $\mathbb{G}_1^\ell$, and $x, e_1, \ldots, e_q$ are uniform in in $\mathbb{Z}_p$, and i^* is uniform in $[q]$.

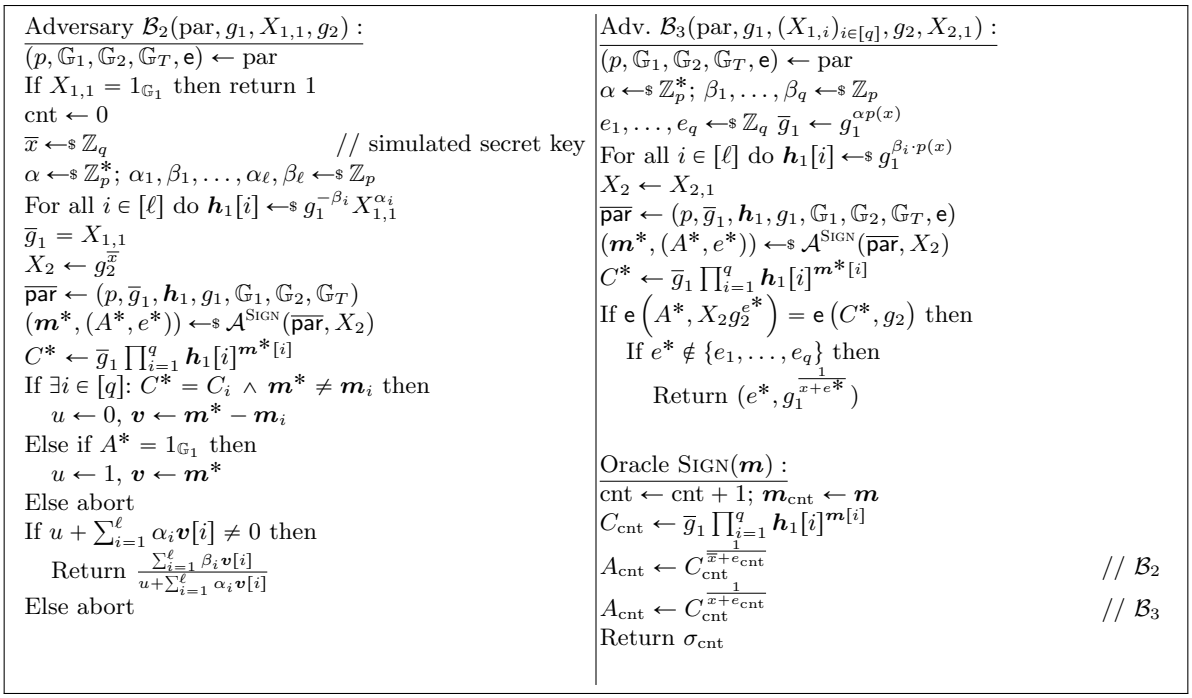

Fig. 5. Adversaries $\mathcal{B}_2$ and $\mathcal{B}_3$ in the proof of Lemmas 2 and 3. In $\mathcal{B}_3$, $p(\mathsf{X}) = \prod_{i\in[q]}(\mathsf{X} + e_i)$. The description of SIGN is shared by both adversaries, but note that the exact computation differs. In $\mathcal{B}_2$, the value of $\overline{x}$ needed to compute A_{cnt} is known to the reduction. In $\mathcal{B}_3$, the value x is used instead, which is not known to the reduction. But $\mathrm{DL}_{g_1}(C_{\mathrm{cnt}})$ is divisible by $(x + e_{\mathrm{cnt}})$, and thus A_{cnt} can be computed.

3.6 Proof of Lemma 2

The argument here is rather standard, and relies on the fact that if Forge_2 occurs, then $\mathcal{A}$ has found messages $\boldsymbol{m} \neq \boldsymbol{m}^*$ such that $\prod_{i=1}^\ell \boldsymbol{h}_1[i]^{(\boldsymbol{m}^*[i]-\boldsymbol{m}[i])} = 1_{\mathbb{G}_1}$, and this in turn allows us to find the discrete logarithm. The case where $A^* = 1_{\mathbb{G}}$ is equivalent to the case where $g_1 \prod_{i=1}^\ell \boldsymbol{h}_1[i]^{\boldsymbol{m}^*[i]} = 1_{\mathbb{G}_1}$

The formal description of $\mathcal{B}_2$ is in Figure 5, on the left. There, q is a bound on the number of queries by $\mathcal{B}$. Given a DL-instance $(g_1, X_{1,1} = g_1^x, g_2)$, the adversary $\mathcal{B}_2$ immediately checks whether $X_{1,1} = 1_{\mathbb{G}_1}$, in which case it returns 0. Otherwise, the adversary simulates $\overline{g}_1 = X_{1,1}$ and $\boldsymbol{h}_1[i] = g_1^{\alpha_i x - \beta_i}$ for $\alpha_i, \beta_i \leftarrow\!\$\ \mathbb{Z}_p$, and produces a secret key value $\overline{x} \leftarrow\!\$\ \mathbb{Z}_p$. Therefore, in both situations, the adversary $\mathcal{B}$ can produce values $u \in \mathbb{Z}_p$, $\boldsymbol{v} \in \mathbb{Z}_p^\ell$, not all zero, such that $\overline{g}_1^u \prod_{i=1}^\ell \boldsymbol{h}_1[i]^{\boldsymbol{v}[i]} = 1_{\mathbb{G}_1}$, or equivalently

$$ux + \sum_{i=1}^{\ell} \boldsymbol{v}[i](\alpha_i x - \beta_i) \ ,$$

or, equivalently, if $u + \sum_{i=1}^\ell \alpha_i \boldsymbol{v}[i] \neq 0$,

$$x = \frac{\sum_{i=1}^\ell \beta_i \boldsymbol{v}[i]}{u + \sum_{i=1}^\ell \alpha_i \boldsymbol{v}[i]} \ ,$$

Note that when $x \neq 0$, the simulation of Game $\mathsf{SUF}_{\mathsf{BBS}}^{\mathcal{A}}(\mathcal{A}, \lambda)$ is perfect, and if Forge_2 occurs, then $(u, \boldsymbol{v})$ are set to some value, and $\mathcal{B}_2$ does not abort. The only way it could possibly abort is if

later $u + \sum_{i=1}^{\ell} \alpha_i \boldsymbol{v}[i] = 0$, and event we denote as Bad, but it is easy to see that $\Pr[\mathsf{Bad}] \leqslant \frac{1}{p}$, independently of the choice of $u, \boldsymbol{v}$. Thus,

$$\begin{aligned}\mathsf{Adv}^{\mathsf{dl}}_{\mathsf{GGen}}(\mathcal{B}_2) &\geqslant \Pr[x = 0] \cdot 1 + \Pr[x \neq 0] \cdot \Pr\left[\mathsf{Forge}_2 \wedge \overline{\mathsf{Bad}}\right] \\ &= \frac{1}{p} + \left(1 - \frac{1}{p}\right) \cdot \Pr[\mathsf{Forge}_2] - \Pr[\mathsf{Bad}] \geqslant \Pr[\mathsf{Forge}_2] - \frac{1}{p} \,,\end{aligned}$$

which concludes the proof.

3.7 Proof of Lemma 3

The description of $\mathcal{B}_3$ is in Figure 5, on the right. Here, we use the fact that given $p(\mathsf{X}) = \prod_{i \in [q]} (\mathsf{X} + e_i)$, we can efficiently compute, for a random $\alpha \leftarrow\!\!\$\, \mathbb{Z}_p^*$, a new generator $\overline{g}_1 = g_1^{\alpha \cdot p(x)}$ given $g_1, X_{1,1} = g_1^x, X_{1,2} = g_2^{x^2}, \ldots, X_{1,q} = g_1^{x^q}$, since $p(\mathsf{X})$ has degree q. If $p(x) \neq 0$, then the simulation of $\overline{g}_1$ has the right distribution. Similarly, we simulate $\boldsymbol{h}_1$ so that $\boldsymbol{h}_1[i] = g_1^{\beta_i p(x)}$. This allows us to easily respond to signature queries, since the discrete logarithm of C_{cnt} is divisible by $x + e_{\mathrm{cnt}}$, and thus A_{cnt} can be computed efficiently. It can also happen that $x = -e_i$ when answering the i-th query, in which case it is easy to answer the query as well, and q-SDH is broken right away. (We do not make this case explicit in the pseudocode.) If $\mathcal{A}$ then indeed outputs a valid forgery for $e^* \notin \{e_1, \ldots, e_q\}$ and $A^* \neq 1_{\mathbb{G}_1}$ (the case $A^* = 1_{\mathbb{G}_1}$ is handled by Forge_2 above), we have

$$\mathrm{DL}_{g_1}(A^*) = \frac{p(x) \cdot \left[\alpha + \sum_{i=1}^{\ell} \beta_i \boldsymbol{m}^*[i]\right]}{x + e^*} \neq 0 \,,$$

because $p(x) \neq 0$, $\alpha + \sum_{i=1}^{\ell} \beta_i \boldsymbol{m}^*[i] \neq 0$, and $x + e^* \neq 0$. We can therefore go ahead and use Remark 1 to find $g_1^{\frac{1}{x+e^*}}$, and break q-SDH.

Because the simulation is perfect when $x \notin \{-e_1, \ldots, -e_q\}$, let GoodE be the event that this is the case, i.e., $p(x) \neq 0$. We then have

$$\begin{aligned}\mathsf{Adv}^{q\text{-}\mathsf{sdh}}_{\mathsf{GGen}}(\mathcal{B}_3) &\geqslant \Pr[\mathsf{Forge}_3 \wedge \mathsf{GoodE}] \\ &\geqslant \Pr[\mathsf{Forge}_3] - \Pr\left[\overline{\mathsf{GoodE}}\right] \geqslant \Pr[\mathsf{Forge}_3] - \frac{q}{p} \,,\end{aligned}$$

which concludes the proof.

4 Tighter Proofs for BBS in the AGM

This section complements the above standard-model analysis with a tight analysis of BBS in the *algebraic group model* (AGM) [FKL18]. In addition, we prove here that security holds even if the attacker is given the values $e_1, e_2, \ldots$ *ahead of time*, and we allow these values to be sampled from a more general distribution. The former fact will be helpful later in Section 6. The latter fact will allow for instantiations of BBS with shorter signatures in some contexts, as we explain further below.

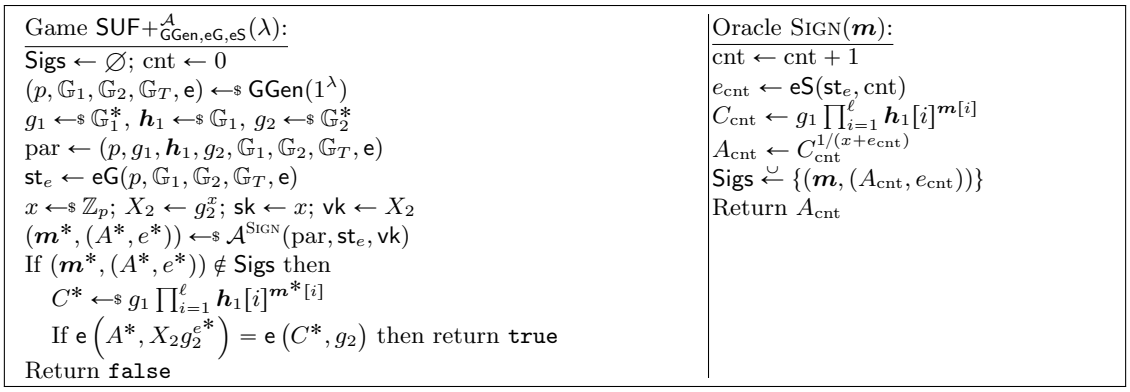

Fig. 6. Stronger security for BBS. Stronger ad-hoc unforgeability achieved by BBS in the AGM, where the e_i's are sampled deterministically from an algorithm that uses an initially generated state st_e, known to the adversary.

Stronger security. We formalize our security goal in terms of Game SUF+ in Figure 6. This is *not* a generic security game, as it is specific to BBS, but clearly, it does imply its strong unforgeability in a number of settings when the scheme instantiation corresponds to a particular pick to eG and eS. The ad-hoc feature is that we allow part of the signature (namely, the e value in a pair (A, e)) to be generated initially. To model this, in addition to the group parameter generator GGen, the game is parameterized by a pair of algorithms, eG and eS, where eG, on input the group parameters, outputs a state st_e, and then $\mathsf{eS}(\mathsf{st}_e, i)$ (deterministically) outputs the value e_i used for the i-th signature. The initial state st_e is given to the adversary, who can run eS to pre-compute the e_i's. It will be convenient to define the collision probability

$$\delta_{\mathsf{eG},\mathsf{eS}}(q, \lambda) = \mathsf{Pr}\left[\begin{array}{l}\mathrm{par} \leftarrow\!\!\$\ \mathsf{GGen}(1^\lambda) \\ \mathsf{st}_e \leftarrow\!\!\$\ \mathsf{eG}(\mathrm{par})\end{array} \ : \ \exists 1 \leqslant i < j \leqslant q : \mathsf{eS}(\mathsf{st}_e, i) = \mathsf{eS}(\mathsf{st}_e, j)\right] .$$

We also define the advantage metric

$$\mathsf{Adv}^{\mathsf{suf+}}_{\mathsf{GGen},\mathsf{eG},\mathsf{eS}}(\mathcal{A}, \lambda) = \mathsf{Pr}\left[\mathsf{SUF{+}}^{\mathcal{A}}_{\mathsf{GGen},\mathsf{eG},\mathsf{eS}}(\lambda)\right] .$$

Algebraic security. We are now ready to state our main theorem, which is proved below in Section 4.1. We dispense with a full formalization of the AGM [FKL18], as its use is relatively straightforward here. Namely, we consider *algebraic adversaries* that provide an explanation of the element $A^* \in \mathbb{G}_1$ contained in the forgery in terms of all previously seen group elements in $\mathbb{G}_1$, which include the generators $g_1, \boldsymbol{h}_1$, as well as the issued signatures. (Because we consider type-3 pairings, we do not include $\mathbb{G}_2$ elements in these representations.) We also give our reduction here to q-DL, as opposed to q-SDH as in the case of Theorem 1, but note that the assumptions are equivalent in the AGM.

Theorem 2 (AGM Security of BBS). *Let* GGen *be a group parameter generator, producing groups of order $p(\lambda)$, and let* eG, eS *as above. For every algebraic SUF+ adversary $\mathcal{A}$ issuing at most q signing queries, there exist adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ such that*

$$\mathsf{Adv}^{\mathsf{suf+}}_{\mathsf{GGen},\mathsf{eG},\mathsf{eS}}(\mathcal{A}, \lambda) \leqslant \mathsf{Adv}^{q\text{-dl}}_{\mathsf{GGen}}(\mathcal{B}_1, \lambda) + \mathsf{Adv}^{\mathsf{dl}}_{\mathsf{GGen}}(\mathcal{B}_2, \lambda) + \delta_{\mathsf{eG},\mathsf{eS}}(q, \lambda) + \frac{1}{p(\lambda)} .$$

The adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ are given explicitly in the proof, and have running times comparable to that of $\mathcal{A}$. The adversary $\mathcal{B}_1$ need to additionally factor a polynomial of degree (at most) q.

The only property required from eG and eS is that $\delta_{\mathsf{eG},\mathsf{eS}}(q,\lambda)$ is small. We note that the lack of collisions is a necessary condition. Indeed, if we generate two signatures (A,e), (A',e) for messages $\boldsymbol{m}$ and $\boldsymbol{m}'$, respectively, it is easy to verify that $((A\cdot A')^{\frac{1}{2}},e)$ is a signature for $\frac{1}{2}(\boldsymbol{m}+\boldsymbol{m}')$, where $\frac{1}{2}$ is the inverse of 2 mod p.

The above theorem supports the security (in the AGM) of some interesting and natural instantiations of BBS with shorter signatures, which we discuss next.

Counter BBS. One natural instantiation, which we refer to as *Counter BBS*, generates the e_i's from a counter, i.e., $\mathsf{eS}(\mathsf{st}_e,i)=i$. This can be advantageous if the signer can reliably maintain such a counter. Signatures then would consist of a group element in $\mathbb{G}_1$ and then $\log q$ additional bits, where q is an upper bound on the number of issued signatures. In particular, one could safely set $q=2^{50}$ in many applications, leading to very short signatures.

Truncated BBS. A different application scenario considers a conservative instantiation that uses a 384-bit group $\mathbb{G}_1$ to prevents Cheon's attack [Che06]. Then, the above bound allows us to choose the e_i's from $\mathbb{Z}_{2^{256}}$, as opposed to $\mathbb{Z}_p$ for a 384-bit prime p, hence saving 128-bit of signature length. We refer to the resulting scheme as *Truncated BBS*. While we need to rely on the AGM to trust this optimization, we do note that the uniformity of the e_i's needed by Theorem 1 appears to be an artifact of the proof, and does not appear to prevent actual attacks.

4.1 Proof of Theorem 2

Before we turn to the construction of the adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$, and their formal analysis, we introduce the algebraic framework that will guide their construction.

Algebraic framework. To start with, in an execution of $\mathsf{SUF}{+}\mathsf{CDH}^{\mathcal{A}}_{\mathsf{GGen},\mathsf{eG},\mathsf{eS}}(\lambda)$, it is convenient to associate the discrete logarithms of group elements in $\mathbb{G}_1$ with formal rational functions (which are then evaluated in the actual execution to obtain the discrete logarithm). In particular, let us denote the discrete logarithms of $\boldsymbol{h}_1$ to the base g_1 by the vector $\boldsymbol{y}\in\mathbb{Z}_p^\ell$. Then, the i-th Sign query for $\boldsymbol{m}\in\mathbb{Z}_p^\ell$, where $e_i=e$, returns a value with discrete logarithm $\varphi^{\boldsymbol{y}}_{\boldsymbol{m},e}(x)$, where

$$\varphi^{\boldsymbol{y}}_{\boldsymbol{m},e}(\mathsf{X})=\frac{1+\langle\boldsymbol{y},\boldsymbol{m}\rangle}{\mathsf{X}+e_i}\ .$$

Here, $\langle\boldsymbol{x},\boldsymbol{y}\rangle$ denotes inner product in $\mathbb{Z}_p$. It turns out that these functions are essentially linearly independent, except for some unfortunate configurations for $\boldsymbol{y}$. This is captured by the following central lemma.

Lemma 5. *Let $e_1,\ldots,e_q\in\mathbb{Z}_p$ be distinct, let $\boldsymbol{y}\in\mathbb{Z}_p^\ell$, and let $\boldsymbol{m}_1,\ldots,\boldsymbol{m}_q\in\mathbb{Z}_p^\ell$. Further, let $(\boldsymbol{m}^*,e^*)\notin\{(\boldsymbol{m}_i,e_i)\}_{i\in[q]}$. Then, assume that there exist $\lambda_1,\ldots,\lambda_q,\gamma\in\mathbb{Z}_p$ such that*

$$\varphi^{\boldsymbol{y}}_{\boldsymbol{m}^*,e^*}(\mathsf{X})=\sum_{i=1}^{q}\lambda_i\cdot\varphi^{\boldsymbol{y}}_{\boldsymbol{m}_i,e_i}(\mathsf{X})+\gamma\ . \tag{5}$$

Then, one of the following two conditions must be true:

(i) There exists $i\in[q]$ such that $e^*=e_i$ and $1-\lambda_i+\langle\boldsymbol{y},\boldsymbol{m}^*-\lambda_i\cdot\boldsymbol{m}_i\rangle=0$.
(ii) We have $e^*\notin\{e_1,\ldots,e_q\}$, but $1+\langle\boldsymbol{y},\boldsymbol{m}^*\rangle=0$.

Proof. To verify (i), assume indeed that $e^* \in \{e_1, \ldots, e_q\}$, and wlog, let $e^* = e_1$. We multiply both sides of (5) by $p(\mathsf{X}) = \prod_{i=1}^{\ell}(\mathsf{X} + e_i)$, and after re-arranging terms, we get

$$(1 - \lambda_1 + \langle \boldsymbol{y}, \boldsymbol{m}^* - \lambda_1 \cdot \boldsymbol{m}_1 \rangle) \cdot p_1(\mathsf{X}) = \gamma \cdot p(\mathsf{X}) + \sum_{i=2}^{q} \lambda_i \cdot \varphi^{\boldsymbol{y}}_{\boldsymbol{m}_i, e_i}(\mathsf{X}) \cdot p_i(\mathsf{X}) \ , \tag{6}$$

where we have used the shorthand $p_i(\mathsf{X}) = \prod_{j \in [q] \setminus \{i\}}(\mathsf{X} + e_j) = p(\mathsf{X})/(\mathsf{X} + e_i)$. We claim that the LHS and RHS of (6) cannot be identical functions, unless $1 - \lambda_1 + \langle \boldsymbol{y}, \boldsymbol{m}^* - \lambda_1 \cdot \boldsymbol{m}_1 \rangle = 0$. Indeed, the RHS is always divisible by $\mathsf{X} + e_1$, because either $\lambda_2, \ldots, \lambda_q$ are all 0, in which case this is vacuously true, or $p(\mathsf{X})$ and $p_i(\mathsf{X})$ for $i \geqslant 2$ are divisible by $(\mathsf{X} + e_1)$. In contrast, if $1 - \lambda_1 + \langle \boldsymbol{y}, \boldsymbol{m}^* - \lambda_1 \cdot \boldsymbol{m}_1 \rangle \neq 0$, then the RHS is not divisible by $\mathsf{X} + e_1$ because $p_1(e_1) \neq 0$.

Let us consider instead the case $e^* \notin \{e_1, \ldots, e_q\}$. For notational convenience, we let $e_{q+1} = e^*$, $p'(\mathsf{X}) = \prod_{i \in [q+1]}(\mathsf{X} + e_i)$, and $p'_k(\mathsf{X}) = \prod_{i \in [q+1] \setminus \{k\}}(\mathsf{X} + e_i)$. Then, multiplying both sides of (5) by $p'(\mathsf{X})$ yields

$$(1 + \langle \boldsymbol{y}, \boldsymbol{m}^* \rangle) \cdot p'_{q+1}(\mathsf{X}) = \gamma \cdot p'(\mathsf{X}) + \sum_{i=2}^{q} \lambda_i \cdot \varphi^{\boldsymbol{y}}_{\boldsymbol{m}_i, e_i}(\mathsf{X}) \cdot p'_i(\mathsf{X}) \ .$$

We notice that if $1 + \langle \boldsymbol{y}, \boldsymbol{m}^* \rangle \neq 0$ the LHS is non-zero, and not divisible by $\mathsf{X} + e_{q+1}$, as $p'_{q+1}(e_{q+1}) \neq 0$. In contrast, the RHS is always divisible by $\mathsf{X} + e_{q+1}$. A contradiction. □

Overview of the reduction. Let $\mathcal{A}$ be an algebraic adversary in Game $\mathsf{SUF{+}{\cdot}}^{\mathcal{A}}_{\mathsf{GGen},\mathsf{eG},\mathsf{eS}}(\lambda)$. It initially receives group elements $g_1 \in \mathbb{G}_1$, $\boldsymbol{h}_1 \in \mathbb{G}_1^{\ell}$, along with $\mathbb{G}_2$ elements $g_2, X_2 = g_2^x$. For each signing query, she also gets $A_i \in \mathbb{G}_1$. Finally, when producing a forgery $(\boldsymbol{m}^*, (A^*, e^*))$, by virtue of being algebraic, the adversary $\mathcal{A}$ also provides a representation $(\gamma_0, \gamma_1, \ldots, \gamma_\ell, \lambda_1, \ldots, \lambda_q) \in \mathbb{Z}_p^{q+\ell+1}$ of A^* such that

$$A^* = g_1^{\gamma_0} \prod_{i=1}^{\ell} \boldsymbol{h}_1[i]^{\gamma_i} \prod_{i=1}^{q} A_i^{\lambda_i} = (C^*)^{\frac{1}{x+e^*}} = g_1^{\varphi^{\boldsymbol{y}}_{\boldsymbol{m}^*, e^*}(x)} \ ,$$

where $\boldsymbol{y}[i] = \mathrm{DL}_{g_1}(\boldsymbol{h}_1[i])$ for all $i \in [\ell]$. Further, we have $A_i = g_1^{\varphi^{\boldsymbol{y}}_{\boldsymbol{m}_i, e_i}(x)}$. Therefore, setting $\gamma = \gamma_0 + \sum_{i \in [\ell]} \gamma_i \cdot \boldsymbol{y}[i]$, this implies in particular that

$$\gamma + \sum_{i=1}^{q} \lambda_i \cdot \varphi^{\boldsymbol{y}}_{\boldsymbol{m}_i, e_i}(x) - \varphi^{\boldsymbol{y}}_{\boldsymbol{m}^*, \boldsymbol{e}^*}(x) = 0 \ .$$

Let us now assume that the two conditions in Lemma 5 do not hold, and that $e_1, \ldots, e_q$ are distinct. Then, Lemma 5 implies that

$$\rho(\mathsf{X}) = \gamma + \sum_{i=1}^{q} \lambda_i \cdot \varphi^{\boldsymbol{y}}_{\boldsymbol{m}_i, e_i}(\mathsf{X}) - \varphi^{\boldsymbol{y}}_{\boldsymbol{m}^*, \boldsymbol{e}^*}(\mathsf{X}) \neq 0 \ ,$$

and therefore x is one of its zeros. Assuming that $x \notin \{-e_1, \ldots, -e_q, -e^*\}$, such zeros can be obtained by factoring the non-zero polynomial

$$q(\mathsf{X}) = \rho(\mathsf{X}) \cdot \prod_{e \in \{e_1, \ldots, e_q, e^*\}} (\mathsf{X} + e) \ ,$$

which has degree at most $q + 1$. One of the zeros has to equal x.

We still need to handle the case where either $1 + \langle \boldsymbol{y}, \boldsymbol{m}^* \rangle = 0$ or $1 - \lambda_i + \langle \boldsymbol{y}, \boldsymbol{m}^* - \lambda_i \cdot \boldsymbol{m}_i \rangle = 0$. It is however not hard to see that this gives us non-trivial discrete logarithm relation, and we can use this to compute the discrete logarithm directly.

FORMAL REDUCTION. To formalize the above analysis, we consider three events during the execution of Game $\mathsf{SUF{+}}^{\mathcal{A}}_{\mathsf{GGen},\mathsf{eG},\mathsf{eS}}(\lambda)$:

- Forge: This is the event that $\mathcal{A}$ outputs a successful forgery and wins the game.
- Rel: This is the event that the forgery is for a message $\boldsymbol{m}^*$ such that either Condition (i) or (ii) of Lemma 5 holds.
- Col: Is the event that there exist distinct $i, j \in [q]$ with $e_i = e_j$.

Then, by the law of total probability,

$$\begin{aligned}\mathsf{Adv}^{\mathsf{suf+}}_{\mathsf{GGen},\mathsf{eG},\mathsf{eS}}(\mathcal{A},\lambda) &= \mathsf{Pr}\left[\mathsf{Forge} \wedge \overline{\mathsf{Rel}}\right] + \mathsf{Pr}\left[\mathsf{Forge} \wedge \mathsf{Rel}\right] \\ &\leqslant \mathsf{Pr}\left[\mathsf{Forge} \wedge \overline{\mathsf{Rel}} \wedge \mathsf{Col}\right] + \mathsf{Pr}\left[\mathsf{Forge} \wedge \overline{\mathsf{Rel}} \wedge \overline{\mathsf{Col}}\right] + \mathsf{Pr}\left[\mathsf{Rel}\right] \\ &\leqslant \mathsf{Pr}\left[\mathsf{Col}\right] + \mathsf{Pr}\left[\mathsf{Forge} \wedge \overline{\mathsf{Rel}} \wedge \overline{\mathsf{Col}}\right] + \mathsf{Pr}\left[\mathsf{Rel}\right] \ .\end{aligned}$$

By definition, we know that $\mathsf{Pr}\left[\mathsf{Col}\right] = \delta_{\mathsf{eG},\mathsf{eS}}(q,\lambda)$. We now give adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ such that

$$\mathsf{Pr}\left[\mathsf{Forge} \wedge \overline{\mathsf{Rel}} \wedge \overline{\mathsf{Col}}\right] \leqslant \mathsf{Adv}^{q\text{-}\mathsf{dl}}_{\mathsf{GGen}}(\mathcal{B}_1) \ , \quad \mathsf{Pr}\left[\mathsf{Rel}\right] \leqslant \mathsf{Adv}^{\mathsf{dl}}_{\mathsf{GGen}}(\mathcal{B}_2) + \frac{1}{p} \ .$$

THE ADVERSARY $\mathcal{B}_1$. The formal description of q-DL adversary $\mathcal{B}_1$ is in Figure 7. If $x \notin \{-e_1, \ldots, -e_q\}$ (which can be checked right away, and gives x), the reduction simulates the original generator g_1 as $\overline{g}_1 = g_1^{\alpha p(x)}$, where $p(x) = \prod_{i\in[q]}(x + e_i)$. (This can be computed given the inputs $X_{1,i} = g^{x^i}$ for $i \in [q]$.) Since $\alpha \in \mathbb{Z}_p^*$ and $p(x) \neq 0$, the simulation is perfect. Also, we can easily compute the answers to SIGN queries due to our choice of $\overline{g}_1$. The adversary then checks that $q(\mathsf{X})$ is non-zero (which is implied by $\overline{\mathsf{Rel}}$), and if so, proceeds to compute its zeros. It is easy to verify that the adversary succeeds with probability at least $\mathsf{Pr}\left[\mathsf{Forge} \wedge \overline{\mathsf{Rel}} \wedge \overline{\mathsf{Col}}\right]$.

THE ADVERSARY $\mathcal{B}_2$. The construction is somewhat standard and given in Figure 8. Let us assume one of the two conditions leading to Rel occurs in Game $\mathsf{SUF{+}}^{\mathcal{A}}_{\mathsf{GGen},\mathsf{eG},\mathsf{eS}}(\lambda)$. First, if (i) occurs for some $i \in [q]$, then

$$g_1^{1-\lambda_i} \prod_{j=1}^{\ell} \boldsymbol{h}_1^{\boldsymbol{m}^*[j]-\lambda_i \boldsymbol{m}_i[j]} = 1_{\mathbb{G}_1} \ . \tag{7}$$

In contrast, if (ii) occurs, then

$$g_1 \prod_{j=1}^{\ell} \boldsymbol{h}_1^{\boldsymbol{m}_i[j]} = 1_{\mathbb{G}_1} \ . \tag{8}$$

Therefore, in both cases, we obtain a non-zero vector $(a, \boldsymbol{b}) \in \mathbb{Z}_p^{\ell+1}$ such that $g_1^a \prod_{i\in[\ell]} \boldsymbol{h}_1[i]^{\boldsymbol{b}[i]} = 1_{\mathbb{G}_1}$. Given the DL instance $X_{1,1} = g_1^x \in \mathbb{G}_1$, the adversary $\mathcal{B}_2$ simulates the generator by picking $\alpha_i, \beta_i \leftarrow\!\!\$\, \mathbb{Z}_p$ for $i \in [\ell]$, and lets

$$\overline{g}_1 = X_{1,1} = g_1^x \ , \quad \boldsymbol{h}_1[i] = X_{1,1}^{\alpha_i} g_1^{-\beta_i} = g_1^{\alpha_i x - \beta_i} \quad \text{for } i \in [\ell] \ .$$

This simulates the right distribution if $X_{1,1} \neq 1_{\mathbb{G}_1}$, which is ensured beforehand. Then, as outlined above, the adversary simulates a correct execution with $\mathcal{A}$, and checks if we obtain a non-trivial relation as above. If so, we are given a non-zero $(a, \boldsymbol{b}) \in \mathbb{Z}_p^{\ell+1}$ such that

$$ax + \sum_{i=1}^{\ell} \boldsymbol{b}[i] \cdot (\alpha_i x - \beta_i) = 0 \ ,$$

Adversary $\mathcal{B}_1(\mathrm{par}, g_1, X_{1,1}, X_{1,2}, \ldots, X_{1,q}, g_2, X_{2,1})$:
$(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}) \leftarrow \mathrm{par}$
$\mathsf{st}_e \leftarrow \mathsf{eG}(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e})$
For all $i \in [q]$ do $e_i \leftarrow \mathsf{eS}(\mathsf{st}_e, i)$
If $\exists i \in [q]$: $g_1^{-e_i} = X_{1,1}$ then return $-e_i$
If $\exists i, j \in [q]$: $i \neq j \ \wedge \ e_i = e_j$ then abort
$\mathsf{Sigs} \leftarrow \varnothing$
$\alpha \leftarrow\!\!\$\ \mathbb{Z}_p^*$; $\boldsymbol{y} \leftarrow\!\!\$\ \mathbb{Z}_p^\ell$; $\overline{g}_1 \leftarrow g_1^{\alpha \cdot p(x)}$
For $i = 1$ to ℓ do $\boldsymbol{h}_1[i] \leftarrow \overline{g}_1^{\boldsymbol{y}[i]}$
$X_2 \leftarrow X_{2,2}$; $\overline{\mathsf{par}} \leftarrow (p, \overline{g}_1, \boldsymbol{h}_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e})$
$(\boldsymbol{m}^*, (A^*, e^*), \{\lambda_i\}_{i\in[q]}, \{\gamma_i\}_{i\in[\ell]}, \gamma_0) \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Sign}}(\overline{\mathsf{par}}, \mathsf{st}_e, X_2)$
$C^* \leftarrow \overline{g}_1 \prod_{i=1}^q \boldsymbol{h}_1[i]^{\boldsymbol{m}^*[i]}$
If $\mathsf{e}(A^*, X_2 g_2^{e^*}) = \mathsf{e}\left(C^*, g_2\right)$ and $(A^*, e^*) \notin \mathsf{Sigs}$ then
  $\gamma \leftarrow \gamma_0 + \sum_{i=1}^\ell \gamma_i \boldsymbol{y}[i]$
  $\rho(\mathsf{X}) = -\varphi^{\boldsymbol{y}}_{\boldsymbol{m}^*, e^*}(\mathsf{X}) + \gamma + \sum_{i=1}^q \lambda_i \varphi^{\boldsymbol{y}}_{\boldsymbol{m}_i, e_i}(\mathsf{X})$
  $q(\mathsf{X}) \leftarrow \rho(\mathsf{X}) \cdot \prod_{e\in\{e_1,\ldots,e_q,e^*\}}(\mathsf{X} + e)$
  If $q(\mathsf{X}) = 0$ then abort
  $\mathsf{Z} \leftarrow \{x \in \mathbb{Z}_p \ : \ q_{\boldsymbol{y}}(x) = 0\}$
  Return $x \in \mathsf{Z}$ s.t. $g_1^x = X_{1,1}$

Oracle $\textsc{Sign}(\boldsymbol{m})$:
$\mathrm{cnt} \leftarrow \mathrm{cnt} + 1$, $\boldsymbol{m}_{\mathrm{cnt}} \leftarrow \boldsymbol{m}$
$C_{\mathrm{cnt}} \leftarrow \overline{g}_1 \prod_{i=1}^q \boldsymbol{h}_1[i]^{\boldsymbol{m}[i]}$
$A_{\mathrm{cnt}} \leftarrow C_{\mathrm{cnt}}^{\frac{1}{x+e_{\mathrm{cnt}}}}$
$\sigma_{\mathrm{cnt}} \leftarrow (A_{\mathrm{cnt}}, e_{\mathrm{cnt}})$
$\mathsf{Sigs} \xleftarrow{\cup} \{\sigma_{\mathrm{cnt}}\}$
Return A_{cnt}

Fig. 7. Adversary $\mathcal{B}_1$ in the proof of Theorem 2. Recall that once $e_1, \ldots, e_q$ are fixed and understood from the context, we use the shorthand $p(\mathsf{X}) = \prod_{i\in[q]}(\mathsf{X} + e_i)$ for convenience.

Adversary $\mathcal{B}_2(\mathrm{par}, g_1, X_{1,1}, g_2)$:
$(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}) \leftarrow \mathrm{par}$
If $X_{1,1} = 1_{\mathbb{G}_1}$ then return 1
$\mathsf{st}_e \leftarrow \mathsf{eG}(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e})$
$\mathrm{cnt} \leftarrow 0$
$\overline{x} \leftarrow\!\!\$\ \mathbb{Z}_q$ // simulated secret key
$\alpha_1, \beta_1, \ldots, \alpha_\ell, \beta_\ell \leftarrow\!\!\$\ \mathbb{Z}_p$
For all $i \in [\ell]$ do $\boldsymbol{h}_1[i] \leftarrow\!\!\$\ X_{1,1}^{\alpha_i} g_1^{-\beta_i}$
$\overline{g}_1 = X_{1,1}$; $X_2 \leftarrow g_2^{\overline{x}}$
$\overline{\mathsf{par}} \leftarrow (p, \overline{g}_1, \boldsymbol{h}_1, g_1, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e})$
$(\boldsymbol{m}^*, (A^*, e^*), \{\lambda_i\}_{i\in[q]}, \{\gamma_i\}_{i\in[\ell]}, \gamma_0) \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Sign}}(\overline{\mathsf{par}}, \mathsf{st}_e, X_2)$ $C^* = \overline{g}_1 \prod_{i=1}^\ell \boldsymbol{h}_1[i]^{\boldsymbol{m}^*[i]}$
If $\exists i : e^* = e_i \ \wedge \ C^* \cdot C_i^{-\lambda_i} = 1_{\mathbb{G}_1}$ then
  $a \leftarrow 1 - \lambda_i$; $\boldsymbol{b} \leftarrow \boldsymbol{m}^* - \lambda_i \boldsymbol{m}_i$
else if $C^* = 1_{\mathbb{G}}$ then
  $a \leftarrow 1$; $\boldsymbol{b} \leftarrow \boldsymbol{m}^*$
If $a + \sum_{i=1}^\ell \alpha_i \boldsymbol{b}[i] = 0$ then abort
Return $\frac{\sum_{i=1}^\ell \beta_i \boldsymbol{b}[i]}{a + \sum_{i=1}^\ell \alpha_i \boldsymbol{b}[i]}$

Oracle $\textsc{Sign}(\boldsymbol{m})$:
$\mathrm{cnt} \leftarrow \mathrm{cnt} + 1$, $\boldsymbol{m}_{\mathrm{cnt}} \leftarrow \boldsymbol{m}$
$e_{\mathrm{cnt}} \leftarrow \mathsf{eS}(\mathsf{st}_e, \mathrm{cnt})$
$C_{\mathrm{cnt}} \leftarrow \overline{g}_1 \prod_{i=1}^\ell \boldsymbol{h}_1[i]^{\boldsymbol{m}[i]}$
$A_{\mathrm{cnt}} \leftarrow C_{\mathrm{cnt}}^{\frac{1}{\overline{x}+e_{\mathrm{cnt}}}}$
$\sigma_{\mathrm{cnt}} \leftarrow (A_{\mathrm{cnt}}, e_{\mathrm{cnt}})$
Return A_{cnt}

Fig. 8. Adversary $\mathcal{B}_2$ in the proof of Theorem 2.

and, in turn, we get

$$x = \frac{\sum_{i=1}^\ell \beta_i \boldsymbol{b}[i]}{a + \sum_{i=1}^\ell \alpha_i \boldsymbol{b}[i]} \ .$$

Note that this is well defined, unless $a + \sum_{i=1}^{\ell} \alpha_i \boldsymbol{b}[i] = 0$. However, because $(a, \boldsymbol{b}) \neq \mathbf{0}$, and the fact that the α_i's are uniform and independent given the adversary's view, this happens with probability at most $1/p$. This concludes the proof. □

5 Efficient Proofs of Knowledge for BBS Signatures

We discuss zero-knowledge proofs of knowledge (zkPoK) of a BBS message-signature pair $(\boldsymbol{m}, \sigma)$ that are shorter than those from [CDL16] adopted by the RFC draft [LKWL22]. If we do not want to reveal k of the components of $\boldsymbol{m}$, our proof (when compiled as a NIZK via the Fiat-Shamir transform) consists of 2 elements in $\mathbb{G}_1$, as well as $k+3$ scalars in $\mathbb{Z}_p$. The prior proof, in contrast, consists of 3 elements in $\mathbb{G}_1$, and $k+5$ scalars. The benefit of our new proofs is also computational: We save 4 group exponentiations in $\mathbb{G}_1$ and 3 scalar multiplications for the prover, and save 3 group exponentiations in $\mathbb{G}_1$ for the verifier. We note that the CDL proofs are tailored at BBS+, but even for the latter scheme, we can achieve savings, as we can think of a BBS+ signature for $\boldsymbol{m}$ as a BBS signature for $(s, \boldsymbol{m})$, for a secret s, and merely increase k by one.

5.1 Proofs of Knowledge for Signatures

We consider zkPoKs associated with a signature scheme SS with message space $\mathsf{SS.M}(\mathrm{par}) = \mathcal{M}(\mathrm{par})^{\ell}$ for some set $\mathcal{M}$ that can depend on the public parameters par, and some understood vector length $\ell = \ell(\lambda)$. We give proofs of knowledge of a signature consistent with a partial message vector $\boldsymbol{m} \in (\mathcal{M} \cup \{\star\})^{\ell}$. For any two such partial messages $\boldsymbol{m}, \boldsymbol{m}'$, we denote $\boldsymbol{m} \subseteq \boldsymbol{m}'$ if for all $i \in [\ell]$, $\boldsymbol{m}[i] \neq \star$ implies $\boldsymbol{m}[i] = \boldsymbol{m}'[i]$.

We only consider three-move public-coin protocols between a *prover* and a *verifier*, described by a tuple $\mathsf{PoK} = (\mathsf{PoK.P}_1, \mathsf{PoK.P}_2, \mathsf{PoK.C}, \mathsf{PoK.V})$. Informally, we think of running the protocol in settings where the parameters for SS are available, i.e., $\mathrm{par} \leftarrow\!\!\$\ \mathsf{SS.Setup}(1^{\lambda})$, $(\mathsf{sk}, \mathsf{vk}) \leftarrow\!\!\$\ \mathsf{SS.KG}(\mathrm{par})$, and the protocol is run as follows, on private input $(\boldsymbol{m}, \sigma)$, where $\boldsymbol{m} \in \mathcal{M}^{\ell}$, and public input $\boldsymbol{m}' \in (\mathcal{M} \cup \{\star\})^{\ell}$:

- **(1)** The prover initially takes inputs par, vk, and a candidate signature-message pair $(\boldsymbol{m}, \sigma)$, and outputs $(a, \mathsf{st_P}) \leftarrow\!\!\$\ \mathsf{PoK.P}_1(\mathrm{par}, \mathsf{vk}, \boldsymbol{m}', (\boldsymbol{m}, \sigma))$. The message a is sent to the verifier.
- **(2)** The verifier outputs $c \leftarrow\!\!\$\ \mathsf{PoK.C}(\mathrm{par}, \mathsf{vk})$, and the *challenge* c is sent to the prover.
- **(3)** The prover outputs $s \leftarrow\!\!\$\ \mathsf{PoK.P}_2(\mathsf{st_P}, c)$, and sends s to the verifier.
- **(4)** Finally, the verifier outputs a Boolean value $\mathsf{PoK.V}(\mathrm{par}, \mathsf{vk}, \boldsymbol{m}', a, c, s) \in \{\mathtt{true}, \mathtt{false}\}$.

We say that PoK is correct if, whenever $\mathsf{SS.Ver}(\mathrm{par}, \mathsf{vk}, (\boldsymbol{m}, \sigma)) = \mathtt{true}$ and $\boldsymbol{m}' \subseteq \boldsymbol{m}$, then the verifier also outputs $\mathtt{true}$.

Special Soundness. We target *special soundness*. To this end, we say that $(\mathrm{par}, \mathsf{vk}, \boldsymbol{m}', a, c, s)$ is an *accepting* transcript if par is a valid output of $\mathsf{SS.Setup}$, vk is a valid output of $\mathsf{SS.KG}(\mathrm{par})$, c is a valid output of $\mathsf{PoK.C}(\mathrm{par}, \mathsf{vk})$, and $\mathsf{PoK.V}(\mathrm{par}, \mathsf{vk}, \boldsymbol{m}', a, c, s)$ is $\mathtt{true}$.

Definition 2. *We say that* PoK *as above is* special-sound *if there exists an efficient algorithm* $\mathsf{Extract}$ *which, given any two valid transcripts* $(\mathrm{par}, \mathsf{vk}, \boldsymbol{m}', a, c, s)$, $(\mathrm{par}, \mathsf{vk}, \boldsymbol{m}', a, c', s')$ *such that* $c \neq c'$, *then* $(\boldsymbol{m}, \sigma) \leftarrow \mathsf{Extract}(\mathrm{par}, \mathsf{vk}, a, (c, s), (c', s'))$ *is such that* $\mathsf{SS.Ver}(\mathrm{par}, \mathsf{vk}, (\boldsymbol{m}, \sigma)) = \mathtt{true}$ *and* $\boldsymbol{m}' \subseteq \boldsymbol{m}$.

We do not specify more general soundness goals further, as the use of special soundness will largely depend on the concrete security game modeling the security of the system using the PoK.

Distribution $\mathsf{Real}^{\mathcal{A}}_{\mathsf{PoK},\mathsf{SS}}(\lambda)$:	Distribution $\mathsf{Ideal}^{\mathcal{A},\mathcal{S},\mathcal{L}}_{\mathsf{PoK},\mathsf{SS}}(\lambda)$:
$\mathsf{par} \leftarrow\!\!\$\ \mathsf{SS.Setup}(1^\lambda)$	$\mathsf{par} \leftarrow\!\!\$\ \mathsf{SS.Setup}(1^\lambda)$
$(\mathsf{sk},\mathsf{vk}) \leftarrow\!\!\$\ \mathsf{SS.KG}(\mathrm{par})$	$(\mathsf{sk},\mathsf{vk}) \leftarrow\!\!\$\ \mathsf{SS.KG}(\mathrm{par})$
$(\boldsymbol{m}', \boldsymbol{m}, \sigma) \leftarrow\!\!\$\ \mathcal{A}(\mathrm{par}, \mathsf{sk}, \mathsf{vk})$	$(\boldsymbol{m}', \boldsymbol{m}, \sigma) \leftarrow\!\!\$\ \mathcal{A}(\mathrm{par}, \mathsf{sk}, \mathsf{vk})$
If $\boldsymbol{m}' \subseteq \boldsymbol{m} \wedge \mathsf{SS.Ver}(\mathrm{par}, \mathsf{vk}, (\boldsymbol{m}, \sigma))$ then	If $\boldsymbol{m}' \subseteq \boldsymbol{m} \wedge \mathsf{SS.Ver}(\mathrm{par}, \mathsf{vk}, (\boldsymbol{m}, \sigma))$ then
$\quad (a, \mathsf{st}_\mathsf{P}) \leftarrow\!\!\$\ \mathsf{PoK.P}_1(\mathrm{par}, \mathsf{vk}, \boldsymbol{m}', (\boldsymbol{m}, \sigma))$	$\quad z \leftarrow\!\!\$\ \mathcal{L}(\mathrm{par}, \mathsf{sk})$
$\quad c \leftarrow\!\!\$\ \mathsf{PoK.C}(\mathrm{par}, \mathsf{vk})$	$\quad (a, c, s) \leftarrow\!\!\$\ \mathcal{S}(\mathrm{par}, \mathsf{vk}, \boldsymbol{m}', z)$
$\quad s \leftarrow\!\!\$\ \mathsf{PoK.P}_2(\mathsf{st}_\mathsf{P}, c)$	$\quad$ Return $(\mathsf{sk}, \mathsf{vk}, \boldsymbol{m}', (a, c, s))$
$\quad$ Return $(\mathsf{sk}, \mathsf{vk}, \boldsymbol{m}', (a, c, s))$	Return $\perp$
Return $\perp$	

Fig. 9. Distributions for the definition of HVZK

Honest-Verifier Zero-Knowledge. The protocols we give will be shown to be honest-verifier zero-knowledge, which suffices for their use as NIZKs via the Fiat-Shamir transform. We will in fact weaken the notion to allow for some *leakage* of the parameters given to the simulator. In particular, we model such leakage as a (possibly randomized) function $\mathcal{L}(\mathsf{par}, \mathsf{sk})$ taking as input the parameters and the signing key.

Definition 3 (HVZK). *The protocol* PoK *for* SS *as above is perfectly* $\mathcal{L}$-honest-verifier zero-knowledge *($\mathcal{L}$-HVZK) if there exists an efficient simulator* $\mathcal{S}$ *such that for all* $\mathcal{A}$ *and* $\lambda \in \mathbb{N}$*, the distributions* $\mathsf{Real}^{\mathcal{A}}_{\mathsf{PoK},\mathsf{SS}}(\lambda)$ *and* $\mathsf{Ideal}^{\mathcal{A},\mathcal{S},\mathcal{L}}_{\mathsf{PoK},\mathsf{SS}}(\lambda)$ *given in Figure 9 are identical.*

5.2 Protocols

Full disclosure. We start with the protocol for the case $\boldsymbol{m} = \boldsymbol{m}'$, i.e., the full-disclosure case. Recall that a BBS signature for a message $\boldsymbol{m} \in \mathbb{Z}_p^\ell$ takes form $\sigma = (A = C(\boldsymbol{m})^{1/(x+e)}, e)$, where $C(\boldsymbol{m}) = g_1 \prod_{i=1}^{\ell} \boldsymbol{h}_1[i]^{\boldsymbol{m}[i]}$. We assume from now on that $A \neq 1_{\mathbb{G}_1}$, and this assumption is almost without loss of generality, as a valid signature with $A = 1_{\mathbb{G}_1}$ implies finding a non-trivial DLOG relation, as $C(\boldsymbol{m}) = 1_{\mathbb{G}_1}$ would be true as well. Recall that the signature is valid if

$$\mathsf{e}(A, g_2^e X_2) = \mathsf{e}(C(\boldsymbol{m}), g_2) \ ,$$

where $X_2 = g_2^x$ is the verification key. However, by bilinearity,

$$\mathsf{e}(A, g_2^e X_2) = \mathsf{e}(A, g_2^e) \cdot \mathsf{e}(A, X_2) = \mathsf{e}(A^e, g_2) \cdot \mathsf{e}(A, X_2) \ .$$

And therefore, one can equivalently check that

$$\mathsf{e}(A, X_2) = \mathsf{e}(C(\boldsymbol{m}) \cdot A^{-e}, g_2) \ . \tag{9}$$

The main idea is to provide suitably randomized versions of A and $B = C(\boldsymbol{m}) \cdot A^{-e}$, which can be computed from a signature (A, e), and extend this with a proof of correctness attesting to the format of these values. In particular, we use $\overline{A} = A^r$ and $\overline{B} = (C(\boldsymbol{m}) \cdot A^{-e})^r$, for which we still have $e(\overline{A}, X_2) = e(\overline{B}, g_2)$.

PROTOCOL DESCRIPTION. Concretely, we consider the following Σ-protocol:

- Given a signature (A, e) for the message $\boldsymbol{m}$ with $A \neq 1_{\mathbb{G}_1}$, the prover picks $r \leftarrow \mathbb{Z}_p^*$, and computes
$$\overline{A} \leftarrow A^r \ , \quad \overline{B} \leftarrow (C(\boldsymbol{m})A^{-e})^r = C(\boldsymbol{m})^r \overline{A}^{-e} \ .$$
It also picks $\alpha, \beta \leftarrow\!\!\$\ \mathbb{Z}_p$, and computes $U \leftarrow C(\boldsymbol{m})^\alpha \overline{A}^\beta$. It sends $(\overline{A}, \overline{B}, U)$ to the verifier.
- The verifier picks a random challenge $c \leftarrow\!\!\$\ \mathbb{Z}_p$, and sends it to the prover
- The prover responds with (s, t), where
$$s \leftarrow \alpha + r \cdot c \ , \ \ t \leftarrow \beta - e \cdot c \ .$$
- The verifier accepts if and only if
$$\mathsf{e}(\overline{A}, X_2) = \mathsf{e}(\overline{B}, g_2) \ , \quad U \cdot \overline{B}^c = C(\boldsymbol{m})^s \overline{A}^t \ .$$

SPECIAL-SOUNDNESS. It is not hard to see that the protocol is special sound. Indeed, given $\overline{A}, \overline{B}, U$, as well as $c_1 \neq c_2$, and (s_1, s_1, t_1, t_2) such that
$$\mathsf{e}(\overline{A}, X_2) = \mathsf{e}(\overline{B}, g_2) \ , \quad U \cdot \overline{B}^{c_i} = C(\boldsymbol{m})^{t_i} \overline{A}^{s_i} \quad \text{for } i = 1, 2 \ ,$$
we can first extract r and e such that $\overline{B} = C(\boldsymbol{m})^r \overline{A}^{-e}$, because
$$\overline{B}^{c_1 - c_2} = C(\boldsymbol{m})^{t_1 - t_2} \overline{A}^{s_1 - s_2} \ ,$$
and thus we can set $r = (t_1 - t_2)/(c_1 - c_2)$ and $e = (s_2 - s_1)/(c_1 - c_2)$. If $r \neq 0$, then $(A = \overline{A}^{r^{-1}}, e)$ is a valid signature on $\boldsymbol{m}$, because $\mathsf{e}(\overline{A}, X_2) = \mathsf{e}(\overline{B}, g_2)$ implies that $\mathsf{e}(\overline{A}^{r^{-1}}, X_2) = \mathsf{e}(\overline{B}^{r^{-1}}, g_2)$, and $\overline{B}^{r^{-1}} = C(\boldsymbol{m})A^{-e}$. If $r = 0$, then $e(\overline{A}, X_2) = e(\overline{A}^{-e}, g_2)$, which means $x = -e$, and this gives us a signature on $\boldsymbol{m}$.

ZERO-KNOWLEDGE. The protocol is $\mathcal{L}$-HVZK, for $\mathcal{L}$ which, on input g_1, x, outputs (g_1^r, g_1^{rx}) for $r \leftarrow\!\!\$\ \mathbb{Z}_p^*$, i.e., a random pair of form (U, U^x). The simulator then computes $\overline{A} \leftarrow\!\!\$\ \mathbb{G}_1^*$, and set $\overline{C} = \overline{A}^x \in \mathbb{G}_1$ – this can be done by re-randomizing the leakage (U, U^x). Then, the simulator picks a random challenge $c \leftarrow\!\!\$\ \mathbb{Z}_p$, as well as random $s, t \leftarrow\!\!\$\ \mathbb{Z}_p$, and sets $U = C(\boldsymbol{m})^s \overline{A}^t \overline{B}^{-c}$.

The fact that the simulator needs a sample (U, U^x), and cannot simulate solely given the parameters and the verification key $X_2 = g_2^x$ is a technical oddity inherited from the use of type-3 pairings, and was also present in prior protocols [CDL16]. Indeed, it is hard to compute g_1^x from the verification key g_2^x. However, this additional leakage is not really harmful. For example, *any* signature (A, e) on a message $\boldsymbol{m}$ already satisfies $A^x = C(\boldsymbol{m}) \cdot A^{-e}$, and thus the protocol leaks no more than *any* valid message-signature pair. In particular, BBS remains secure given such leakage.

PARTIAL DISCLOSURE. For the case $\boldsymbol{m}' \subsetneq \boldsymbol{m}$, the components $\boldsymbol{m}[i]$ for which $\boldsymbol{m}'[i] = \star$ become parts of the witness. We let $I := \{i \in [\ell] \ : \ \boldsymbol{m}[i] = \star\}$ and $J = [n] \backslash I$. We also let $C_J(\boldsymbol{m}) = g_1 \prod_{i \in J} \boldsymbol{h}_1[i]^{\boldsymbol{m}[i]}$, and note that $C_J(\boldsymbol{m})$ can be computed from the public input $\boldsymbol{m}'$ by the verifier.

- Given a signature (A, e) for the message $\boldsymbol{m}$ with $A \neq 1_{\mathbb{G}_1}$, the prover picks $r \leftarrow \mathbb{Z}_p^*$, and computes
$$\overline{A} \leftarrow A^r \ , \quad \overline{B} \leftarrow (C(\boldsymbol{m})A^{-e})^r = C_J(\boldsymbol{m})^r \cdot \left(\prod_{i \in I} \boldsymbol{h}_1[i]^{\boldsymbol{m}[i]} \right)^r \cdot \overline{A}^{-e} \ .$$

It also picks $\alpha, \beta \leftarrow\!\!\$\ \mathbb{Z}_p$, and also $\delta_i \leftarrow\!\!\$\ \mathbb{Z}_p$ for every $i \in I$ and computes

$$U \leftarrow C_J(\boldsymbol{m})^{\alpha} \cdot \overline{A}^{\beta} \cdot \prod_{i \in I} \boldsymbol{h}_1[i]^{\delta_i}$$

It sends $(\overline{A}, \overline{B}, U)$ to the verifier.

- The verifier picks a random challenge $c \leftarrow\!\!\$\ \mathbb{Z}_p$, and sends it to the prover
- The prover responds with $(s, t, (u_i)_{i \in I})$, where

$$s \leftarrow \alpha + r \cdot c\ ,\quad t \leftarrow \beta - e \cdot c\ ,\quad u_i \leftarrow \delta_i + r \cdot \boldsymbol{m}[i] \cdot c\ \ \forall i \in I\ .$$

- The verifier accepts if and only if

$$\mathsf{e}(\overline{A}, X_2) = \mathsf{e}(\overline{B}, g_2)\ ,\quad U \cdot \overline{B}^{c} = C_J(\boldsymbol{m})^{s} \overline{A}^{t} \prod_{i \in I} \boldsymbol{h}_1[i]^{u_i}\ .$$

One can easily adapt the arguments for the above protocols for full disclosure to show special soundness and $\mathcal{L}$-HVZK.

NIZKs. Our protocols can be transformed into NIZKs in the random oracle model via the Fiat-Shamir transform [FS87] or Fischlin's transform [Fis05]. For the Fiat-Shamir version, the prover computes $\overline{A}, \overline{B}, U$ as above, then lets $c \leftarrow H(\boldsymbol{m}', \overline{A}, \overline{B}, U)$, and finally computes $s, t, (u_i)_{i \in I}$ as above. The final proof is

$$\pi = (\overline{A}, \overline{B}, c, s, t, (u_i)_{i \in I})\ .$$

Verification checks that $\mathsf{e}(\overline{A}, X_2) = \mathsf{e}(\overline{B}, g_2)$ and that $c = H(\boldsymbol{m}', \overline{A}, \overline{B}, U)$ with

$$U \leftarrow \overline{B}^{-c} C_J(\boldsymbol{m})^{s} \overline{A}^{t} \prod_{i \in I} \boldsymbol{h}_1[i]^{u_i}\ .$$

Note that we could include U instead of c, but this leads to longer proofs for curves like BLS12-381, where elements in $\mathbb{G}_1$ have longer descriptions than scalars.

6 Signatures for Group Elements and Blind Issuance

One central property of BBS is its support of *blind issuance*, the setting where a user sends a commitment $C \in \mathbb{G}_1$ to the signer to obtain a pair $\sigma = (A, e)$ with $A = C^{\frac{1}{x+e}}$—if $C = g_1 \prod_{i=1}^{\ell} \boldsymbol{h}_1[i]^{\boldsymbol{m}[i]}$ for a message $\boldsymbol{m}$, then σ is a valid signature on $\boldsymbol{m}$, but crucially, the signer never learns $\boldsymbol{m}$. In fact, the user could make $\boldsymbol{m}[1]$ uniform, turning C into a perfectly-hiding (generalized) Pedersen commitment [Ped92]. This approach is particularly important when σ acts as a credential, and we want to hide the actual attributes from the issuer. Blind issuance of BBS signatures is also part of an unofficial draft [Bli], which also requires the addition of a proof of knowledge for a representation of C, which consists of $O(\ell)$ scalars and can be expensive when ℓ is large. Here, we show that in the AGM the scheme is already sufficiently secure without such a proof. A suitable proof of knowledge is however still necessary if the user needs to reveal part of the attributes to the issuer, to prove these are consistent with the commitment. However, we note that this aspect would be orthogonal to our analysis below.

Game $\mathsf{OMUF}{+}\mathcal{A}_{\mathsf{GGen},\mathsf{eG},\mathsf{eS}}(\lambda)$:
$\mathrm{cnt} \leftarrow 0$
$(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}) \leftarrow\!\$\ \mathsf{GGen}(1^\lambda)$
$g_1 \leftarrow\!\$\ \mathbb{G}_1^*,\ \boldsymbol{h} \leftarrow\!\$\ \mathbb{G}_1,\ g_2 \leftarrow\!\$\ \mathbb{G}_2^*$
$\mathrm{par} \leftarrow (p, g_1, \boldsymbol{h}_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e})$
$\mathsf{st}_e \leftarrow \mathsf{eG}(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e})$
$x \leftarrow\!\$\ \mathbb{Z}_p;\ X_2 \leftarrow g_2^x;\ \mathsf{sk} \leftarrow x;\ \mathsf{vk} \leftarrow X_2$
$\{(\boldsymbol{m}_i^*, (A_i^*, e_i^*))\}_{i\in[q']} \leftarrow\!\$\ \mathcal{A}^{\textsc{Sdh}}(\mathrm{par}, \mathsf{st}_e, \mathsf{vk})$
For all $i \in [q']$ do $C_i^* \leftarrow g_1 \prod_{j\in[\ell]} \boldsymbol{h}_1[j]^{\boldsymbol{m}_i^*[j]}$
If $\forall i \in [q']$: $\mathsf{e}\left(A_i^*, X_2 g_2^{e_i^*}\right) = \mathsf{e}\left(C_i^*, g_2\right)$ then
  Return $(q' > \mathrm{cnt})$
Return `false`

Oracle $\textsc{Sdh}(C)$:
$\mathrm{cnt} \leftarrow \mathrm{cnt} + 1$
$e_{\mathrm{cnt}} \leftarrow \mathsf{eS}(\mathsf{st}_e, \mathrm{cnt})$
$A_{\mathrm{cnt}} \leftarrow\!\$\ C^{\frac{1}{x+e_{\mathrm{cnt}}}}$
Return A_{cnt}

Fig. 10. One-more unforgeability of BBS. This game captures the one-more unforgeability of BBS when given an SDH oracle which returns $C^{1/(x+e_i)}$ for its i-th query, where e_i is generated via eS. We assume here that $\mathcal{A}$ returns a set of q' distinct forgery attempts (i.e, no double entry are present in the list returned by $\mathcal{A}$.)

ONE-MORE UNFORGEABILITY. BBS can be thought as a signature scheme signing a group element $C \in \mathbb{G}_1$ as $\sigma = (A = C^{\frac{1}{x+e}}, e)$. However, it does not achieve unforgeability when signing group elements (as in the case of *structure-preserving signatures* (SPS) [AFG+10]). Indeed, the attacker, given $\sigma = (A, e)$, directly obtains other valid signatures, such as $\sigma' = (A^2, e)$, which is a valid signature for $C^2 \neq C$. Nonetheless, if $C = g_1 \prod_{i=1}^{\ell} \boldsymbol{h}_1[i]^{\boldsymbol{m}[i]}$, it is very unlikely that the attacker can exhibit a message $\boldsymbol{m}'$ such that $C^2 = g_1 \prod_{i=1}^{\ell} \boldsymbol{h}_1[i]^{\boldsymbol{m}'[i]}$, i.e., such that (A^2, e) is valid for $\boldsymbol{m}'$.

We formalize this by showing BBS satisfies *one-more unforgeability* (OMUF), where given access q times to an oracle SDH that signs group elements as above—i.e., on input $C \in \mathbb{G}_1$ it returns $\sigma = (A, e)$ with $A = C^{\frac{1}{x+e}}$—it is impossible for the attacker to come up with $q+1$ valid BBS signatures. This property is defined via Game $\mathsf{OMUF}{+}\mathcal{A}_{\mathsf{GGen},\mathsf{eG},\mathsf{eS}}(\lambda)$ in Figure 10. Similar to Section 4, the game is parameterized by the group generator GGen and by a pair of algorithms eG, eS used to generate the e_i's ahead of time. We also define

$$\mathsf{Adv}^{\mathsf{omuf}+}_{\mathsf{GGen},\mathsf{eG},\mathsf{eS}}(\mathcal{A}, \lambda) = \Pr\left[\mathsf{OMUF}{+}\mathcal{A}_{\mathsf{GGen},\mathsf{eG},\mathsf{eS}}(\lambda)\right] .$$

We stress that we *could* define a general notion of signatures on commitment values, and require that upon obtaining q signatures on arbitrary elements from the commitment space, the attacker cannot come up with $q+1$ valid signatures on commitments, along with their openings. However, we prefer the rather straightforward BBS-specific game as a better illustration of this property.

MAIN RESULT. We prove now that BBS satisfies one-more unforgeability in the AGM, and we do so via a reduction to its SUF+ security as defined in Section 4.

Theorem 3 (One-more unforgeability). *Let* GGen *be a group parameter generator, producing groups of order* $p(\lambda)$*, and let* eG, eS *as above. For every algebraic OMUF+ adversary* $\mathcal{A}$ *issuing at most* $q = q(\lambda)$ SDH *queries, there exists an algebraic SUF+ adversary* $\mathcal{B}$ *such that*

$$\mathsf{Adv}^{\mathsf{omuf}+}_{\mathsf{GGen},\mathsf{eG},\mathsf{eS}}(\mathcal{A}, \lambda) \leqslant \mathsf{Adv}^{\mathsf{suf}+}_{\mathsf{GGen},\mathsf{eG},\mathsf{eS}}(\mathcal{B}, \lambda) + \delta_{\mathsf{eG},\mathsf{eS}}(q, \lambda) .$$

The adversary $\mathcal{B}$ *issues* q SIGN *queries, and runs in time equal that of running* $\mathcal{A}$*, plus the time needed to perform* $O(q^3)$ *operations in* $\mathbb{Z}_p$*.*

The proof is given below. The main challenge in the proof is to show how the signing oracle can be used to simulate signing a group element, given its representation. This is easy to do if the representation is only in terms of g_1 and $\boldsymbol{h}_1$, but the challenge is that the representation can also depend on prior signatures.

Theorem 3 yields the following corollary when combined with Theorem 2.

Corollary 1 (One-more unforgeability). *Let* GGen *be a group parameter generator, producing groups of order* $p(\lambda)$*, and let* eG, eS *as above. For every algebraic OMUF+ adversary* $\mathcal{A}$ *issuing at most* q SDH *queries, there exists a* q*-DL adversary* $\mathcal{C}_1$ *and a DL adversary* $\mathcal{C}_2$ *such that*

$$\mathsf{Adv}^{\mathsf{omuf+}}_{\mathsf{GGen},\mathsf{eG},\mathsf{eS}}(\mathcal{A},\lambda) \leqslant \mathsf{Adv}^{q\text{-}\mathsf{dl}}_{\mathsf{GGen}}(\mathcal{C}_1,\lambda) + \mathsf{Adv}^{\mathsf{dl}}_{\mathsf{GGen}}(\mathcal{C}_2,\lambda) + 2\delta_{\mathsf{eG},\mathsf{eS}}(q,\lambda) + \frac{1}{p(\lambda)} \ .$$

The adversaries $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *are obtained by using* $\mathcal{B}$ *from Theorem 3 within the adversaries of Theorem 2.*

Proof (of Theorem 3). Recall that the adversary $\mathcal{B}$ has access, beforehand, to all values $e_1,\ldots,e_q$ such that the i-th query to SIGN uses the i-th value e_i. This is because it receives st_e, and $e_i = \mathsf{eS}(\mathsf{st}_e, i)$. We will use this fact in a crucial way below. The adversary $\mathcal{B}$ simulates an execution of $\mathsf{OMUF{+}}^{\mathcal{A}}_{\mathsf{GGen},\mathsf{eG},\mathsf{eS}}(\lambda)$ with help of the SIGN oracle. Initially, $\mathcal{A}$ is fed the same parameters $\mathsf{par} = (p, g_1, \boldsymbol{h}_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e})$, st_e, and verification key $X_2 = g_2^x$ given to $\mathcal{B}$. Consequently, $\mathcal{B}$ has use e_i to simulate the answer the i-the SDH query by $\mathcal{A}$

In particular, the adversary $\mathcal{A}$ queries the SDH oracle repeatedly on inputs $C_1, C_2, \ldots, C_q \in \mathbb{G}_1$. (Here, we assume without loss of generality that $\mathcal{A}$ makes *exactly* q queries.) For each of these SDH query, we show next how $\mathcal{B}$ can answer $A_i = C_i^{\frac{1}{x+e_i}}$ using the SIGN oracle and the algebraic representation output by the algebraic adversary $\mathcal{A}$.

THE SIMULATION STRATEGY. An important observation is that $\mathcal{B}$ can easily use its i-th query to SIGN oracle to compute $\overline{A}_i = \overline{C}_i^{\frac{1}{x+e_i}}$ for any $\overline{C}_i \in \mathbb{G}_1$ for which the representation only depends on g_1 and $\boldsymbol{h}_1$. Indeed, if

$$\overline{C}_i = g_1^{\gamma_i} \prod_{j=1}^{\ell} \boldsymbol{h}_1[j]^{\boldsymbol{\alpha}_i[j]} \ ,$$

we can query the message $\boldsymbol{m}_i = \gamma_i^{-1}\boldsymbol{\alpha}_i$, and raise the result to the γ_i to obtain $\overline{A}_i$. The main challenge is that, in general $\mathcal{A}$'s query C_i will come with a representation not only in terms of g_1, $\boldsymbol{h}_1$, but also, in terms all prior queries $A_1,\ldots,A_{i-1}$. This will make computing A_i far more challenging.

What we show next is that given $\mathcal{A}$'s SDH queries $C_1, C_2, \ldots, C_q$, and their representations, we can compute, efficiently, and in an online fashion, a sequence $(\overline{C}_1, \overline{A}_1, \boldsymbol{\lambda}_1), \ldots, (\overline{C}_q, \overline{A}_q, \boldsymbol{\lambda}_q)$ such that the following properties are satisfied:

1. $\overline{A}_i = \overline{C}_i^{\frac{1}{x+e_i}}$
2. We can compute efficiently a representation of $\overline{C}_i$ only in terms of g_1 and $\boldsymbol{h}_1$, and thus, $\overline{A}_i$ can be computed via a query to SIGN from the representation of $\overline{C}_i$.
3. $\boldsymbol{\lambda}_i \in \mathbb{Z}_p^i$, and $A_i = C_i^{\frac{1}{x+e_i}}$ satisfies $A_i = \prod_{j=1}^{i} \overline{A}_i^{\boldsymbol{\lambda}_i[j]}$.

Clearly, $\mathcal{B}$ will exactly compute, after receiving each query C_i, the corresponding $\overline{C}_i$, to then obtain $\overline{A}_i$, and thus also A_i. (We discuss how to produce a forgery later on.) We show by induction, now, how to produce the sequence.

BUILDING THE SEQUENCE. It is easy to obtain $\overline{A}_1, \overline{C}_1$. Indeed, $\mathcal{A}$ provides a representation of C_1 only in terms of g_1 and $\boldsymbol{h}_1$, as there are no prior queries. Then, we can just set $\overline{C}_1 = C_1$, and $\overline{A}_1 = A_1$ can be obtained by $\mathcal{B}$ via a query to the SIGN oracle. Consequently, $\boldsymbol{\lambda}_1[1] = 1$.

Now, let us assume we have obtained such a sequence up to the i-th query, i.e., we have $(\overline{C}_1, \overline{A}_1, \boldsymbol{\lambda}_1), \ldots, (\overline{C}_i, \overline{A}_i, \boldsymbol{\lambda}_i)$ as above. By property (3) above, the representation of the $(i+1)$-st SDH query C_{i+1}, which is originally in terms of $g_1, \boldsymbol{h}_1, A_1, \ldots, A_i$, can be rewritten as

$$C_{i+1} = g_1^{\gamma_{i+1}} \prod_{j=1}^{\ell} \boldsymbol{h}_1[j]^{\boldsymbol{\alpha}_{i+1}[j]} \prod_{j=1}^{i} A_i^{\boldsymbol{\beta}_i[j]} = g_1^{\gamma_{i+1}} \prod_{j=1}^{\ell} \boldsymbol{h}_1[j]^{\boldsymbol{\alpha}_{i+1}[j]} \prod_{j=1}^{i} \overline{A}_j^{\boldsymbol{\rho}_i[j]}$$

for some $\boldsymbol{\rho} \in \mathbb{Z}_p^i$. We now need to compute $C_{i+1}^{\frac{1}{x+e_{i+1}}}$, and to this end, for all $j \in [i]$, the main challenge is to compute $\overline{A}_j^{\frac{\boldsymbol{\rho}_i[j]}{x+e_{i+1}}}$. To this end, assume that $e_{i+1} \neq e_j$, and first observe that

$$\begin{aligned}
\overline{A}_j^{\frac{\boldsymbol{\rho}_i[j]}{x+e_{i+1}}} &= \overline{C}_j^{\frac{\boldsymbol{\rho}_i[j]}{(x+e_j)(x+e_{i+1})}} \\
&= \overline{C}_j^{\frac{\boldsymbol{\rho}_i[j]}{e_{i+1}-e_j}\left(\frac{1}{x+e_j} - \frac{1}{x+e_{i+1}}\right)} \\
&= \overline{A}_j^{\frac{\boldsymbol{\rho}_i[j]}{e_{i+1}-e_j}} \cdot \left(\overline{C}_j^{\frac{\boldsymbol{\rho}_i[j]}{e_j-e_{i+1}}}\right)^{\frac{1}{x+e_{i+1}}} .
\end{aligned}$$

Therefore, if $e_{i+1} \notin \{e_1, \ldots, e_i\}$,

$$\begin{aligned}
A_{i+1} &= C_{i+1}^{\frac{1}{x+e_{i+1}}} \\
&= \left(g_1^{\gamma_{i+1}} \prod_{j=1}^{\ell} \boldsymbol{h}_1[j]^{\boldsymbol{\alpha}_{i+1}[j]}\right)^{\frac{1}{x+e_{i+1}}} \prod_{j=1}^{i} \overline{A}_j^{\frac{\boldsymbol{\rho}_i[j]}{e_{i+1}-e_j}} \cdot \left(\overline{C}_j^{\frac{\boldsymbol{\rho}_i[j]}{e_j-e_{i+1}}}\right)^{\frac{1}{x+e_{i+1}}} \\
&= \overline{C}_{i+1}^{\frac{1}{x+e_{i+1}}} \cdot \prod_{j=1}^{i} \overline{A}_j^{\frac{\boldsymbol{\rho}_i[j]}{e_{i+1}-e_j}} ,
\end{aligned}$$

where

$$\overline{C}_{i+1} = g_1^{\gamma_{i+1}} \cdot \prod_{j=1}^{\ell} \boldsymbol{h}_1[j]^{\boldsymbol{\alpha}_{i+1}[j]} \cdot \prod_{j=1}^{i} \overline{C}_j^{\frac{\boldsymbol{\rho}_i[j]}{e_j-e_{i+1}}} ,$$

for which we can obtain a representation that only depends on g_1 and $\boldsymbol{h}_1$ by the fact that such a representation is computed for $\overline{C}_1, \ldots, \overline{C}_i$. Then, we set $\overline{A}_{i+1} = \overline{C}_{i+1}^{\frac{1}{x+e_{i+1}}}$. In conclusion,

$$A_{i+1} = \overline{A}_{i+1} \cdot \prod_{j=1}^{i} \overline{A}_j^{\frac{\boldsymbol{\rho}_i[j]}{e_{i+1}-e_j}} ,$$

and this allows us to set $\boldsymbol{\lambda}_{i+1}[j] = \frac{\boldsymbol{\rho}_i[j]}{e_{i+1}-e_j}$ for all $j \in [i]$, and $\boldsymbol{\lambda}_{i+1}[i+1] = 1$.

Outputting the final forgery. Finally, the adversary $\mathcal{A}$ outputs a sequence of signatures

$$(\boldsymbol{m}_1^*, (A_1^*, e_1^*)), \ldots, (\boldsymbol{m}_{q'}^*, (A_{q'}^*, e_{q'}^*)) ,$$

along with representation of all A_i^* in terms of $g_1, \boldsymbol{h}_1$, and $A_1, \ldots, A_q$, where $q' > q$. Assume the adversary indeed wins, i.e., these signatures are all valid. Then, by the pigeonhole principle, there exists $k \in [q']$ such that $(A_k^*, e_k^*) \neq (\widetilde{A}_i, e_i)$ for all $i \in [q]$, where $\widetilde{A}_i$ is the (normalized) value returned by the Sign query used to compute $\overline{A}_i$, and therefore $\mathcal{B}$ outputs $(\boldsymbol{m}_k^*, (A_k^*, e_k^*))$ as its valid forgery.

Note that we require $\mathcal{B}$ to be also algebraic, and for this, it suffices to give a representation of A_k^* in terms of $\widetilde{A}_1, \ldots, \widetilde{A}_q$, g_1 and $\boldsymbol{h}_1$. This can be done easily using the available representation—while it is in terms of $A_1, \ldots, A_q$, we can use property (3) above to turn it into a suitable representation.

Probability analysis. The above simulation strategy is perfect as long as $e_1, \ldots, e_q$ are distinct, and therefore, given the event Forge that $\mathcal{A}$ outputs q' valid signatures for $q' > q$, and let Coll be the event that two of the e_i's collide. Then,

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{suf}+}_{\mathsf{GGen},\mathsf{eG},\mathsf{eS}}(\mathcal{B}, \lambda) &\geqslant \mathsf{Pr}\left[\mathsf{Forge} \wedge \overline{\mathsf{Coll}}\right] \\
&\geqslant \mathsf{Pr}\left[\mathsf{Forge}\right] - \mathsf{Pr}\left[\mathsf{Coll}\right] \\
&= \mathsf{Adv}^{\mathsf{omuf}+}_{\mathsf{GGen},\mathsf{eG},\mathsf{eS}}(\mathcal{A}, \lambda) - \delta_{\mathsf{eG},\mathsf{eS}}(q, \lambda) .
\end{aligned}$$

Efficiency consideration. The main cost of $\mathcal{B}$ stems from maintaining representations of the A_i's in terms of the $\boldsymbol{A}_i$. These representation are i-dimensional vectors, with i as large as q. Therefore, computing $\overline{C}_{i+1}$ can cost up to $O(q^2)$ operations in $\mathbb{Z}_p$ per oracle query. □

Applications. As mentioned above, a typical application of BBS signatures is in the context of credentials. The above result validates the security of the canonical solution where the user obtains a credential for a vector of attributes $\boldsymbol{m}$ by sending $C = g_1 \prod_{i=1}^{\ell} \boldsymbol{h}_1[i]^{\boldsymbol{m}[i]}$ to the authority, which in turn responds with the actual credential $(C^{\frac{1}{x+e}}, e)$ for a random e. The OMUF security from Theorem 3 and Corollary 1 implies that a malicious user (or any set of multiple such users) can only obtain q credentials by interacting with the authority q times. The user can then show the credential multiple times in an unlinkable way by using the zk-PoKs from Section 5, typically compiled via the Fiat-Shamir transform. These showings are then consistent with at most q attribute vectors. When issuing a credential, the user does not need to send any proof of knowledge along with C, unless the credential issuing needs to enforce some format on the values contained by C, in which case extra proofs need to be sent along.

We note that analyzing the security of the entire credential system is non-trivial, especially if we want to resort to PoKs compiled via the Fiat-Shamir transform, which are not online extractable. We believe that a security analysis of variants of this system is however possible, albeit very tedious, in the AGM, where one can resort to the online-extractability of the proposed PoKs from Section 5 in the AGM, along the lines of [GT21]. This goes however beyond the scope of this paper.

Acknowledgments

We wish to thank Christian Paquin and Greg Zaverucha for extensive discussions around BBS and for providing feedback throughout this project. We also thank the EUROCRYPT 2023 reviewers for their excellent comments and suggestions. This research was partially supported by NSF grants CNS-2026774, CNS-2154174, a JP Morgan Faculty Award, a CISCO Faculty Award, and a gift from Microsoft.

References

AFG+10. Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, Heidelberg, August 2010.

ASM06. Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic k-TAA. In Roberto De Prisco and Moti Yung, editors, *SCN 06*, volume 4116 of *LNCS*, pages 111–125. Springer, Heidelberg, September 2006.

BB08. Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008.

BBSa. BBS+ implementation. https://github.com/mattrglobal/bbs-signatures. Accessed: 2022-10-04.

BBSb. BBS+ implementation. https://github.com/microsoft/bbs-node-reference. Accessed: 2022-10-04.

BBS04. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Heidelberg, August 2004.

BFL20. Balthazar Bauer, Georg Fuchsbauer, and Julian Loss. A classification of computational assumptions in the algebraic group model. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 121–151. Springer, Heidelberg, August 2020.

BG04. Daniel R. L. Brown and Robert P. Gallant. The static Diffie-Hellman problem. Cryptology ePrint Archive, Report 2004/306, 2004. https://eprint.iacr.org/2004/306.

BL10. Ernie Brickell and Jiangtao Li. A pairing-based DAA scheme further reducing TPM resources. In Alessandro Acquisti, Sean W. Smith, and Ahmad-Reza Sadeghi, editors, *Trust and Trustworthy Computing, Third International Conference, TRUST 2010, Berlin, Germany, June 21-23, 2010. Proceedings*, volume 6101 of *Lecture Notes in Computer Science*, pages 181–195. Springer, 2010.

BL11. Ernie Brickell and Jiangtao Li. Enhanced privacy ID from bilinear pairing for hardware authentication and attestation. *Int. J. Inf. Priv. Secur. Integr.*, 1(1):3–33, 2011.

Bli. Blind signatures extension of the BBS signature scheme. https://identity.foundation/bbs-signature/draft-blind-bbs-signatures.txt. Accessed: 2022-10-04.

BLS03. Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 257–267. Springer, Heidelberg, September 2003.

Bow17. Sean Bowe. BLS12-381: New zk-SNARK elliptic curve construction. https://electriccoin.co/blog/new-snark-curve/, 2017.

CDL16. Jan Camenisch, Manu Drijvers, and Anja Lehmann. Anonymous attestation using the strong diffie hellman assumption revisited. In Michael Franz and Panos Papadimitratos, editors, *Trust and Trustworthy Computing - 9th International Conference, TRUST 2016, Vienna, Austria, August 29-30, 2016, Proceedings*, volume 9824 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2016.

Che06. Jung Hee Cheon. Security analysis of the strong Diffie-Hellman problem. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 1–11. Springer, Heidelberg, May / June 2006.

Che09. Liqun Chen. A DAA scheme requiring less TPM resources. In Feng Bao, Moti Yung, Dongdai Lin, and Jiwu Jing, editors, *Information Security and Cryptology - 5th International Conference, Inscrypt 2009, Beijing, China, December 12-15, 2009. Revised Selected Papers*, volume 6151 of *Lecture Notes in Computer Science*, pages 350–365. Springer, 2009.

CL03. Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 268–289. Springer, Heidelberg, September 2003.

CL04. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, Heidelberg, August 2004.

CS14. Shan Chen and John P. Steinberger. Tight security bounds for key-alternating ciphers. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 327–350. Springer, Heidelberg, May 2014.

Fis05. Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 152–168. Springer, Heidelberg, August 2005.

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.

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.

GPS06. S.D. Galbraith, K.G. Paterson, and N.P. Smart. Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165, 2006. https://eprint.iacr.org/2006/165.

GT21. Ashrujit Ghoshal and Stefano Tessaro. Tight state-restoration soundness in the algebraic group model. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 64–93, Virtual Event, August 2021. Springer, Heidelberg.

HT16. Viet Tung Hoang and Stefano Tessaro. Key-alternating ciphers and key-length extension: Exact bounds and multi-user security. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 3–32. Springer, Heidelberg, August 2016.

JY09. David Jao and Kayo Yoshida. Boneh-Boyen signatures and the strong Diffie-Hellman problem. In Hovav Shacham and Brent Waters, editors, *PAIRING 2009*, volume 5671 of *LNCS*, pages 1–16. Springer, Heidelberg, August 2009.

LKWL22. Tobias Looker, Vasilis Kalos, Andrew Whitehead, and Mike Lodder. The BBS Signature Scheme. Internet-Draft draft-irtf-cfrg-bbs-signatures-01, Internet Engineering Task Force, October 2022. Work in Progress.

Mau09. Ueli M. Maurer. Unifying zero-knowledge proofs of knowledge. In Bart Preneel, editor, *AFRICACRYPT 09*, volume 5580 of *LNCS*, pages 272–286. Springer, Heidelberg, June 2009.

Pat08. Jacques Patarin. A proof of security in $O(2^n)$ for the Benes scheme. In Serge Vaudenay, editor, *AFRICACRYPT 08*, volume 5023 of *LNCS*, pages 209–220. Springer, Heidelberg, June 2008.

Ped92. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992.

PS16. David Pointcheval and Olivier Sanders. Short randomizable signatures. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 111–126. Springer, Heidelberg, February / March 2016.

SAB+19. Alberto Sonnino, Mustafa Al-Bassam, Shehar Bano, Sarah Meiklejohn, and George Danezis. Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers. In *NDSS 2019*. The Internet Society, February 2019.

zca. Cheon's attack and its effect on the security of big trusted setups. https://ethresear.ch/t/cheons-attack-and-its-effect-on-the-security-of-big-trusted-setups/6692. Accessed: 2022-10-04.