Multi-Holder Anonymous Credentials from BBS Signatures

Andrea Flamini[1,2], Eysa Lee[3], and Anna Lysyanskaya[4]

[1] Department of Mathematical Sciences, Politecnico di Torino, Torino, Italy
[2] Mathematics Department, University of Trento, Povo, Trento, Italy
andrea.flamini@polito.it
[3] Data Science Institute, Brown University, Providence, RI, USA
eysa_lee@brown.edu
[4] Computer Science Department, Brown University, Providence RI, USA
anna_lysyanskaya@brown.edu

Abstract. The eIDAS 2.0 regulation aims to develop interoperable digital identities for European citizens, and it has recently become law. One of its requirements is that credentials be unlinkable. Anonymous credentials (AC) allow holders to prove statements about their identity in a way that does not require to reveal their identity and does not enable linking different usages of the same credential. As a result, they are likely to become the technology that provides digital identity for Europeans. Any digital credential system, including anonymous credentials, needs to be secured against identity theft and fraud. In this work, we introduce the notion of a *multi-holder anonymous credential scheme* that allows issuing shares of credentials to different authentication factors (or "holders"). To present the credential, the user's authentication factors jointly run a threshold presentation protocol. Our definition of security requires that the scheme provide unforgeability: the adversary cannot succeed in presenting a credential with identity attributes that do not correspond to an identity for which the adversary controls at least t shares; this is true even if the adversary can obtain credentials of its choice and cause concurrent executions of the presentation protocol. Further, our definition requires that the presentation protocol provide security with identifiable abort. Finally, presentations generated by all honest holders must be unlinkable and must not reveal the user's secret identity attributes even to an adversary that controls some of the user's authentication factors.
We design and prove the (concurrent) security of a multi-holder version of the BBS anonymous credential scheme. In our construction, each holder is issued a secret share of a BBS credential. Using these shares, the holders jointly compute a credential presentation that is identical to (and therefore compatible with) the traditional, single-holder variant (due to Tessaro and Zhu, Eurocrypt'23) of a BBS credential presentation.

Table of Contents

1 Introduction

According to W3C Verifiable Credential Data Model[5], "a verifiable credential is a tamper-evident credential that has authorship that can be cryptographically verified". Verifiable credentials are issued by *issuers* to *holders*, and the holders can use them to create *presentations* used to prove claims about their identity to *verifiers*.

Anonymous credentials are a special kind of verifiable credentials and allow a holder to obtain and prove possession of a credential to a verifier in a way that does not require the holder to reveal its identity or the credential itself. This technology is particularly useful to protect the privacy of the holders by preventing the issuers and the verifiers to track the holder's activity.

Anonymous credentials recently attracted renewed interest due to the publication of the eIDAS 2.0 regulation[6], which aims to facilitate secure cross-border transactions by establishing a framework for digital identity and authentication for digital services in the EU. The cryptographic community was invited to provide feedback on this regulation, and the resulting feedback document [BBC+24] recommends the creation of the EUDI wallet (the digital wallet that European citizens will use to store their credential) which might support the use of anonymous credentials; it specifically encourages the EU to use the BBS-based family [BBS04,CL04,ASM06,BL10,CDL16,TZ23,LKWL22] of constructions of anonymous credentials.

At a minimum, anonymous credentials satisfy two main properties, namely *unforgeability* and *privacy*. Unforgeability guarantees that a user cannot generate a verifying presentation without the consent of the issuer, and privacy guarantees that verifiers cannot correlate presentations of the same credential or learn anything about its attributes not explicitly revealed in the presentation. A useful additional property we consider is selective disclosure, which allows the credential holder to choose a subset of signed attributes to reveal to the verifier during the credential presentation phase [FSS+24].

A natural framework for constructing anonymous credentials, the so-called CL framework proposed by Camenisch and Lysyanskaya [CL03], is instantiated in several anonymous credentials systems such as [CL01,CL04,CDL16,PS16,TZ23]. In the CL framework, a credential is a signature on a set of attributes, and to prove possession of the credential, the holder proves in zero-knowledge that they hold a signature on a set of attributes that verifies under the credential issuer's public key.

BBS *Signatures as Anonymous Credentials.* Boneh, Boyen and Shacham [BBS04] gave a group signature scheme that Camenisch and Lysyanskaya [CL04] suggested could be adapted to anonymous credentials. The resulting schemes, BBS and a variant called BBS+, were subsequently analyzed, improved, and adapted, in a provably secure fashion, for use in direct anonymous attestation (DAA) and anonymous credential schemes [ASM06,BL10,CDL16].

[5] https://www.w3.org/TR/vc-data-model-2.0/
[6] https://digital-strategy.ec.europa.eu/en/policies/eidas-regulation

The state-of-the-art security proof for this use of BBS and a zero-knowledge protocol for proving knowledge of a BBS signature were given by Tessaro and Zhu [TZ23]. The BBS signature as described in [TZ23] is the most efficient of the known candidate signatures in the CL framework [CL03,CL04,PS16,CDL16][7] and is also the object of a standardization effort of W3C [LKWL22][8].

Motivation. Digital credentials require that the users protect the cryptographic material representing the credentials. Corruption, loss, or theft of the device where this material is stored can result in identity theft and fraud, defeating the purpose of a digital credential system. For anonymous credentials, the threat is all the more serious here, as it is impossible to trace how the adversary used a stolen credential (unlike in linkable verifiable credentials [AAM23]). Additionally, an adversary who compromises a single-factor credential learns sensitive information about this user, which is a threat to privacy.

Multi-factor authentication is a popular way to enhance the security of digital authentication. For anonymous credentials, it would amount to storing shares of credentials on multiple devices and proving possession of the credential in a distributed fashion. This is similar to how shares of secret keys are used in threshold signature schemes. In particular, if an adversary corrupts at most $t-1$ devices (and therefore learns the value of $t-1$ shares of a credential), it should not be able to generate a valid credential presentation. On the other hand, if a threshold t of the devices agree to present the credential, they can generate a valid presentation executing a multiparty protocol, while keeping their share of the credential private.

1.1 Our Contribution

In this work, we introduce *multi-holder anonymous credential (MHAC)* schemes. In a MHAC scheme, the credential attributes and the credential itself are not stored on a single device of a single user, but instead are distributed among multiple devices and/or holders. An adversary that gains control of fewer than t devices will be unable to demonstrate possession of the credential or even learn anything about the private attributes. In order to present a credential, devices must jointly convince a verifier that a valid credential is distributed among the parties.

An MHAC scheme addresses the same security goals as a single-holder anonymous credential system: *unforgeability*, which roughly means that the adversary cannot present credential attributes that it was not issued, and *privacy*, which means that a credential presentation reveals nothing other than the intended attribute set and cannot be linked to another presentation of the same credential.

[7] A comparison between [CL03,PS16,CDL16,TZ23] is performed in [FSS+24]

[8] The authors of the specification have updated the credential format from BBS+ to BBS signatures after the publication of [TZ23], however they have decided to adopt an alternative protocol for the creation of the presentation of BBS credentials, which has recently been included in an update of the paper of [TZ23, Appendix B].

Let us go over unforgeability for MHAC in more depth. Suppose an adversary controls fewer than t holders of a credential with attributes $\mathbf{a}$ issued by an honest issuer. Further, suppose that the adversary can query the issuer for new credentials with attributes; let $\mathbf{a}_i$ correspond to credentials from query i. It can participate in computing several *concurrent* presentations of a credential where it controls a subset of the holders, and arbitrarily schedule messages in these presentations. Suppose some attribute a_j (or, more generally, a subset of attributes $(a_{j_1}, \ldots, a_{j_\ell})$) does not appear any of $\mathbf{a}_i$. Then the adversary cannot create a valid presentation of a_j, even if it appears in $\mathbf{a}$.

Moreover, we require that, when the adversary controls fewer than t holders, its participation in a credential presentation results either in a correct output for the honest participants, or in the identification (and, as a result, removal) of at least one of the adversarial holders.

As far as privacy is concerned, we consider two different notions based on what information the adversary already knows. Specifically, we require unlinkability (Definition 8) that applies in the case when the adversary controls the credential verifier but none of the credential holders; here, a simulator creates the adversary's view on input just the attributes revealed as part of credential presentation, and this simulated view is indistinguishable from the real one. Additionally, we require attribute hiding (Definition 9) that applies in the case when the adversary controls fewer than t credential holders involved in presenting the credential. Here, the adversary already knows the identity of the holder devices that computed the credential presentation, so the best we can hope for is that the adversary does not learn anything it does not already know about the credential attributes from participating in credential presentation.

Once we put forth these definitions, we satisfy them with a construction of an efficient MHAC scheme compatible with the BBS anonymous credential scheme described by Tessaro and Zhu in [TZ23]. By "compatible" we mean that the setup and verification are identical, and the MHAC credential shares can be derived from the credential issued in the underlying single-party scheme (here, BBS). We prove that our MHAC scheme satisfies our security definition. Our scheme also allows the holders to selectively disclose some of the attributes included in the credential.

1.2 Our Techniques

First, let us recall BBS anonymous credentials. They require a bilinear pairing $\mathbf{e}$ over groups $\mathbb{G}_1$, $\mathbb{G}_2$ of order q with generators g_1 and g_2, and additional generators $h_1, \ldots, h_m$ for the group $\mathbb{G}_1$. The secret key x for the BBS signature scheme is a random element of $\mathbb{Z}_q$, while the public key is $\mathsf{pk} = g_2^x$.

A BBS signature on the message vector $\mathbf{a} = (a_1, \ldots, a_m)$ is of the form (A, e), where $A = C(\mathbf{a})^{\frac{1}{e+x}}$ and $C(\mathbf{a})$ is a way to encode $\mathbf{a}$: $C(\mathbf{a}) = g_1 \prod_{i=1}^{m} h_i^{a_i}$. The BBS verification algorithm verifies that A was computed correctly by checking that $\mathbf{e}(A, (\mathsf{pk})g_2^e) = \mathbf{e}(C(\mathbf{a}), g_2)$, or, equivalently, that $\mathbf{e}(A, \mathsf{pk}) = \mathbf{e}(B, g_2)$, where $B = C(\mathbf{a})A^{-e}$.

Note that if this equality holds for a given pair A and B, then for any $r \in \mathbb{Z}_q$, it will also hold for $\overline{A} = A^r$ and $\overline{B} = B^r = C(\mathbf{a})^r A^{-re} = C(\mathbf{a})^r \overline{A}^{-e}$. Moreover, given $\overline{A}$ and $\overline{B}$ for which this equality holds, and the values $(\alpha, \beta_1, \ldots, \beta_m, \gamma)$ such that $\overline{B} = g_1^{\alpha}(\prod_{i=1}^{m} h_i^{\beta_i})\overline{A}^{\gamma}$, the message vector $\mathbf{a}$ and the BBS signature on this vector can be recovered as follows: set $r = \alpha$, let $a_i = \beta_i/\alpha$, and let $e = -\gamma$.

As a result, a zero-knowledge proof of knowledge of the message vector $\mathbf{a}$ and a signature (A, e) boils down to (1) picking a random r and computing $\overline{A} = A^r$, a "blinded" version of the value A; (2) computing the corresponding $\overline{B} = B^r$; and (3) proving knowledge of the representation of $\overline{B}$ in bases $g_1, h_1, \ldots, h_m$ and $\overline{A}$. A series of papers [CL04,BL10,CDL16] culminating in the work of Tessaro and Zhu [TZ23] showed that indeed the resulting protocol is a zero-knowledge proof of knowledge of a BBS signature.

Credential secret sharing. How do we secret-share a BBS anonymous credential in such a way that the protocol used to create a presentation is efficient? Is it always possible for a holder to perform a secret sharing of its credential irrespective of the type of BBS credential it is issued?

The more naive approach to distributing a BBS credential $((A, e), \mathbf{a})$ would be to include in the credential an extra attribute that is never revealed and distributed among the holders, basically leading to the distribution of a BBS+ anonymous credential [CDL16]. However, this approach would not take full advantage of the use of the more compact BBS anonymous credentials as described in [TZ23], and restricts the distribution of the anonymous credential to credentials including a random attribute which is never disclosed.

Instead, we describe how to distribute any BBS credential by providing each holder with all the attributes signed in that credential, as this is likely to be the most common application, and does not tie the distribution process to the kind of BBS credential issued. This is done by secret-sharing the value e of the BBS signature and providing every holder with the value A^{-e}. Proving that this distribution of the BBS signature is secure is an unexpectedly tricky task that we address in the security proof of the unforgeability of presentations (in Section 7.5 and more in detail in Appendix F.2, Case A).

Given our basic construction, we enhance it by adding an optional feature: the support of distribution of some *private attributes* $\{a_j\}_{j \in \mathsf{Prv}}$ in $\mathbf{a}$ that are especially sensitive and that we might not want to store in the clear on any device. The remaining attributes in $\mathbf{a}$ ($\{a_j\}_{j \in \mathsf{Pub}}$) and the value A will be known to each credential holder, i.e. they are part of the joint input to all participants. Since we aim to be very flexible about the way the attributes are distributed, we plug this feature onto the basic protocol where the value e is t-out-of-n secret-shared. However, in some circumstances, in particular when a private attribute is *never* revealed, the distribution of e becomes unnecessary.

Given its shares $e^{(i)}, \{a_j^{(i)}\}_{j \in \mathsf{Prv}}$ of e and $\{a_j\}_{j \in \mathsf{Prv}}$ as well as the joint input, each holder participates in a joint computation of the proof of knowledge of $\mathbf{a}$, A and e, while possibly revealing some of the attributes in $\{a_j\}_{j \in \mathsf{Pub}}$.

Our protocol for computing this proof is efficient because the value $D = \left(\prod_{j \in \mathsf{Prv}} h_j^{a_j}\right) A^{-e}$ is (implicitly) provided to all the holders. To be more precise, we give to each holder $\{D_i\}_{i \in [n]}$, with $D_i = \left(\prod_{j \in \mathsf{Prv}} h_j^{a_j^{(i)}}\right) A^{-e^{(i)}}$, from which D can be recovered. While hiding the values of $\{a_j\}_{j \in \mathsf{Prv}}$ and e, D allows them to compute the value $\overline{B}$ as $(C(\mathbf{a})A^{-e})^r$, which is necessary to build the proof of knowledge of a BBS signature. The proof of knowledge can be computed by the holders in a distributed fashion by having each participant prove knowledge of a different factor of $\overline{B}$ depending on its secret shares of e and $\{a_j\}_{j \in \mathsf{Prv}}$.

Proving that distributing e and revealing to every holder D is safe is done via a reduction to discrete logarithm. This reduction receives as input from the DL challenger (g, h), and from the unforgeability adversary a set of attributes $\mathbf{a}$, from which it can compute $C(\mathbf{a}) = g_1 \prod_{i=1}^{m} h_i^{a_i}$. The challenging part in designing the reduction resides in the generation of the values $A, \tilde{A} (= A^e)$ satisfying the conditions: (1) $\log_A \tilde{A} = \log_g h$, as well as (2) $A = C(\mathbf{a})^{\frac{1}{x + \log_g h}}$. Thus, if the adversary succeeds in forging a proof of knowledge of this credential, our reduction solves the discrete logarithm problem.

Access to $\{D_i\}_{i \in [n]}$ is also helpful in achieving the identifiable abort property, which allows identifying a malicious participant who would cause the protocol to generate an invalid presentation. When the holders cooperate in the generation of the proof of knowledge of a representation of $\overline{B}$, each participant $\mathsf{P}_i, i \in S$ proves knowledge of a representation of a factor $\tilde{B}_i$ of $\overline{B} = \prod_{i \in S} \tilde{B}_i$ which can be computed by every other party. Therefore if they generate an invalid proof, their misbehaviour can be detected by verifying each participant proof.

We can also optimize the size of the credential shares, which otherwise would be linear in the number of holders (due to the need to store $\{D_i\}_{i \in [n]}$). Instead, at issue time, each D_i will be signed under a public key used just for this purpose and each holder stores only its own signed D_i. Each holder can then send its signed D_i to others as part of the presentation protocol.

Presentation protocol overview. The presentation protocol executed by the parties $\mathsf{P}_i, i \in S, |S| = t$ instructs a protocol participant, the primary party P_j, to sample a random $r \xleftarrow{\$} \mathbb{Z}_p$ and broadcast it to the other parties in S. Next, each participating holder derives $\overline{A} = A^r$ and $\overline{B} = B^r$ as defined in the presentation protocol described in [TZ23] that we recall at the beginning of this section.

The simplest case for our protocol is when the presentation discloses all the attributes $\{a_i\}_{i \in \mathsf{Pub}}$, i.e. the set of revealed indices is $\mathsf{Rev} = \mathsf{Pub}$. Then the presentation is simply a proof of knowledge of a discrete logarithm representation of $\overline{B}$ with respect to $C(\mathbf{a}') = g_1 \prod_{i \in \mathsf{Rev}} h_i^{a_i}, \{h_i\}_{i \in \mathsf{Prv}}$ and $\overline{A}$, i.e.

$$\overline{B} = C(\mathbf{a}')^r \prod_{i \in \mathsf{Prv}} h_i^{r a_i} \overline{A}^{-e}.$$

Note that the credential shares contain the values $D_i, i \in S$, therefore it is possible for every participant to compute

- $\tilde{B}_j = C(\mathbf{a}')^r D_j^{r \lambda_{S,j}(0)}$, corresponding to the primary party P_j;

– $\tilde{B}_i = D_i^{r\lambda_{S,i}(0)}, \forall i \in S \setminus \{j\}$, corresponding to each other party;

where $\lambda_{S,i}(0)$ is the i-th Lagrange coefficient w.r.t. participating parties S.

Moreover each party $\mathsf{P}_i, i \in S \setminus \{j\}$ knows a representation of $\tilde{B}_i$ w.r.t $\{h_i\}_{i\in\mathsf{Prv}}, \overline{A}$, and P_j knows a representation of $\tilde{B}_j$ w.r.t. $C(\mathbf{a}'), \{h_i\}_{i\in\mathsf{Prv}}, \overline{A}$.

Therefore, since $\overline{B} = \prod_{i\in S} \tilde{B}_i$, we instruct each party $\mathsf{P}_i, i \in S$ to prove knowledge of the corresponding $\tilde{B}_i$ with respect to the aforementioned basis in a coordinated way so that the proof of knowledge can be aggregated. More specifically the parties execute a variant of the threshold Schnorr signature Sparkle [CKM23a] producing in output a proof of knowledge of a representation of $\overline{B}$ w.r.t. $C(\mathbf{a}'), \{h_i\}_{i\in\mathsf{Prv}}, \overline{A}$. We show that this results in a concurrently secure protocol.

1.3 Outline

The rest of the paper is organized as follows. We briefly review related works in Section 2 and preliminaries in Section 3. In Section 4, we define the notion of multi-holder anonymous credentials, and in Section 5 we give the security notions a multi-holder anonymous credential must satisfy. In Section 6, we give the construction of a BBS-based multi-holder anonymous credential, and in Section 7 we prove this scheme secure.

2 Related Works

Distributed computation of zero-knowledge proofs. In [KMR12], the authors describe a framework for distributing the prover side of sigma protocols over multiple parties and provide a general characterization of such protocols defining three different flavors of zero-knowledge. The authors apply their framework to user-centric protocols, for example, the sigma protocol used to prove knowledge of a CL anonymous credential [CL04].

The problem of turning the threshold version of a sigma protocol (similar to [KMR12]) to a non-interactive protocol with respect to the verifier has been studied in [BF24], where the authors determine the properties that the threshold sigma protocol must satisfy to obtain an unforgeable threshold signature against static and active adversaries. In their work the prover side does not require the existence of the combiner since they assume a broadcast channel between the provers.

Our work follows the setting adopted in [BF24], and more generally by threshold digital signatures [CKM23a,DKL+23], since we design a protocol that does not require the interaction between the provers and the verifier.

Therefore, in our security analysis, we do not need to consider the case in which the verifier is malicious and we only focus on specific security notions for the application to anonymous credential systems which are:

– the *unforgeability of the presentations*, meaning that an adversary who corrupts at most $t-1$ holders (i.e. knows $t-1$ shares of credentials) can not forge a presentation;

– the *unlinkability of presentation*, meaning that if the participants to the protocol are honest, the presentation is indistinguishable from a simulated presentation.
– the *unlinkability of private attributes*, meaning that an adversary who corrupts at most $t-1$ holders and passively corrupts the issuer can not distinguish if a credential includes a specific private attribute.
– the *identifiable abort*, meaning that the honest parties can identify a misbehaving participant when a presentation creation fails.

Distributed anonymous credentials. There is a line of works which describes solutions to distribute anonymous credentials on two distinct devices which have distinct computational power or corruption models; for instance [HSS23,HS21] distributing an anonymous credential between a digital wallet on a smart phone and a computationally constrained object such as a smart card. In both cases the involvement of the constrained object in the creation of the presentation is essential, but the amount of operations it must perform does not depend on the size of the credential and of the attributes to disclose, and the authors try to keep it as small as possible. Protocols in which the credential is shared between a device (e.g. smartphone) and a server or a blockchain have also been considered in [LHAT20,MY24].

In our work, we describe a protocol which allows the storage and the presentation of credentials over an arbitrary number of devices, with an arbitrary threshold of them needed to present the credentials. Each party is assumed to have enough computational power to carry out the protocol, and we only require that the adversary can corrupt a number of devices below the specified threshold needed to present the credential.

3 Preliminaries

Notation. Let $[n]$ denote the set $\{1, 2, \ldots, n\}$, and let $x \xleftarrow{\$} S$ denote sampling an element x from a set S uniformly randomly. Let $x \xleftarrow{\$} A(i_1, \ldots, i_n)$ denote that x is the output of the probabilistic algorithm A which takes in input $(i_1, \ldots, i_n)$. Alternatively, we may make explicit the randomness used by A by writing $x \leftarrow A(i_1, \ldots, i_n; R)$. A deterministic protocol V taking in input $(j_1, \ldots, j_m)$ and outputting y is represented as $y \leftarrow V(j_1, \ldots, j_m)$.

Security and Communication Model. We work in the synchronous model against a static adversary that can actively corrupt up to $t-1$ holders in the presentation protocol. We assume point-to-point private communication between the issuer and each holder. For the credential presentation protocol, we assume parties have access to a private, authenticated broadcast channel between the set of parties involved in the credential presentation protocol. Moreover, we assume that each session is identified by a unique session identifier ssid agreed upon by the parties involved in the protocol execution, which is included in each message sent between parties and in broadcasts.

Private broadcast and synchrony are simplifying assumptions to describe a simple t-of-n three-round protocol achieving presentation unlinkability and identifiable abort, but it is possible to loosen these requirements. The private channel is needed to achieve the unlinkability of presentations, and we can remove the broadcast channel using techniques similar to those used in [BLT+24] and in [CKM23b] while preserving the unforgeability of presentations. Removing synchrony is more tricky. Without either synchrony or an honest majority, we cannot achieve identifiable abort or guarantee termination (see [CLOS02,CL17]). However, in the asynchronous setting we can still achieve selective abort, meaning that the adversary can choose which executions produce output. The adversary is not able to produce dishonest presentations in either of these settings.

Bilinear Groups. A bilinear group (or *pairing* group) is a trio of groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{T})$ with an efficient map (or pairing) operation $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_\mathsf{T}$, such that (1) for any $x, \in \mathbb{Z}_p$ and $y \in \mathbb{Z}_p$, $\mathbf{e}(g_1^x, g_2^y) = \mathbf{e}(g_1, g_2)^{x \cdot y}$ and (2) $\mathbf{e}(g_1, g_2) \neq 1$. There are three types of pairings [GPS08]: type-1, in which $\mathbb{G}_1 = \mathbb{G}_2$; type-2, in which $\mathbb{G}_1 \neq \mathbb{G}_2$ and there exists an efficient isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$; and type-3, in which $\mathbb{G}_1 \neq \mathbb{G}_2$ and there does not exist an efficient isomorphism ψ.

Secret Sharing. A classic technique to create a t-of-n secret sharing of a value v is Shamir's secret sharing [Sha79]: a dealer samples a random $(t-1)$-degree polynomial $p(\cdot)$ such that $p(0) = v$ and gives each party P_i their own point on the polynomial $p(i)$. Given at least t points, Lagrange interpolation can be used to reconstruct p and retrieve v. We use $\mathsf{Share}(t, n, v)$ to denote the dealer's algorithm for generating a t-of-n Shamir secret sharing of v. That is, $\{p(i)\}_{i \in [n]} \xleftarrow{\$} \mathsf{Share}(t, n, v)$, where $p(0) = v$. We also make use of *verifiable* secret sharing (VSS), a variant of secret sharing which considers a possibly corrupt dealer who may distribute shares that do not correspond to a valid sharing of a value. VSS allows parties to verify that their received shares correspond to a valid sharing of some value v.

Hardness Assumptions. We recall hardness assumptions BBS and our construction rely on: the discrete logarithm (DL) assumption and the q-strong Diffie-Hellman (qSDH) assumption.

Definition 1 (Discrete logarithm assumption). *Let* $\mathsf{pp} \leftarrow (\mathbb{G}, p, g)$ *where* $\mathbb{G}$ *is a cyclic groups of prime order* p *with generator* g*. The discrete logarithm (*DL*) assumption holds in* $\mathbb{G}$ *if for any PPT adversary* $\mathcal{A}$

$$\Pr[\mathcal{A}(\mathsf{pp}, g, g^x) = x)] \leq \mathsf{negl}(\lambda)$$

where $x \xleftarrow{\$} \mathbb{Z}_p, (g, g^x) \in \mathbb{G}^2$ *and* κ *is the security parameter.*

Definition 2 (q-Strong Diffie-Hellman assumption [BB08]). *Let* $\mathbb{G}_1$ *and* $\mathbb{G}_2$ *be two cyclic groups of prime order* p *with generators* g_1 *and* g_2*, respectively.*

The q-Strong Diffie-Hellman (qSDH*) assumption holds in* $(\mathbb{G}_1, \mathbb{G}_2)$ *if for any PPT adversary* $\mathcal{A}$

$$\Pr[\mathcal{A}(\mathbb{G}_1, \mathbb{G}_2, g_1, \{g_1^{(x^i)}\}_{i\in[q]}, g_2, g_2^x) = (c, g_1^{\frac{1}{x+c}})] \leq \mathsf{negl}(\lambda)$$

where $(g_1, \{g_1^{(x^i)}\}_{i\in[q]}, g_2, g_2^x) \in \mathbb{G}_1^{q+1} \times \mathbb{G}_2^2$ *and* λ *is the security parameter.*

3.1 Sigma protocols

Sigma protocols are a type of proof of knowledge with a three-move structure, where the first message is a commitment from the prover, the second is a random challenge from the verifier, and the final message is a response from the prover. We recall the formal definition of a sigma protocol

in the notation of [Dam02] as follows.

Definition 3 (Sigma protocol [Dam02, Definition 1]). *Given a relation* $\mathcal{R}$*, a sigma protocol* π *for the relation* $\mathcal{R}$ *is an interactive protocol between a prover* P *and a verifier* V *with 3-move form, i.e.* P *sends a commitment* cmt *to* V*, who replies to* P *with a random challenge* ch*, and finally* P *computes a response* rsp *that is sent back to* V*.* π *also satisfies the following properties:*

- π *has* completeness*, which means that if* P *and* V *execute* π *with common input* y *and private input* w *to* P*, with* $(w, y) \in \mathcal{R}$*,* V *always accepts;*
- π *has* special soundness*, which means that, from any* y*, and from a pair of valid conversations for input* $y, (\mathsf{cmt}, \mathsf{ch}, \mathsf{rsp}), (\mathsf{cmt}, \mathsf{ch}', \mathsf{rsp}')$ *with* $\mathsf{ch} \neq \mathsf{ch}'$*, one can easily compute* w *s.t.* $(w, y) \in \mathcal{R}$*;*
- π *has* (special) honest-verifier zero-knowledge (HVZK)*, which means that there exists a polynomial time algorithm* Sim *which on input a statement* y *and a random challenge* ch *outputs a transcript* $(\mathsf{cmt}, \mathsf{ch}, \mathsf{rsp})$ *with the same distribution of the real conversations between honest* P *and* V *on input* y*.*

In later sections we make use of a standard sigma protocol for linear relations, which we recall in Appendix A (Figure 1) for reference.

3.2 BBS signatures

The BBS anonymous credential scheme presented by Tessaro and Zhu [TZ23] is one of the pillars of our work. The authors revisit the security analysis of the BBS signature [BBS04] and provide a novel protocol to prove possession of a credential.

The idea of using BBS signatures [BBS04] to generate anonymous credentials was initially proposed by Camenisch and Lysyanskaya in [CL04, Section 5], and a slightly modified version known as BBS+ was studied and proven unforgeable by [ASM06,CDL16]. [TZ23] later showed the modification is not needed for unforgeability and propose a protocol for proof of possession (which could be applied also to BBS+ signatures) which produces proofs smaller in size.

Definition 4 (BBS **signature scheme [BBS04,CL04]).** *The algorithms defining the* BBS *digital signature are the following:*

- $\mathsf{Pgen}_{\mathsf{BBS}}(\kappa)$. *Let* $\mathbb{G}_1 = \langle g_1 \rangle, \mathbb{G}_2 = \langle g_2 \rangle$ *and* $\mathbb{G}_T$ *be groups of prime order* p*, and* $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ *be the pairing operation. Sample* $h_1, \ldots, h_m \xleftarrow{\$} \mathbb{G}_1$ *and set the set of public parameters* $\mathsf{pp} \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2, h_1, \ldots, h_m)$.
- $\mathsf{KGen}_{\mathsf{BBS}}(\mathsf{pp})$. *Sample a random* $x \xleftarrow{\$} \mathbb{Z}_p$. *Compute* $X_2 = g_2^x$*, and set* $\mathsf{sk} \leftarrow x$*, and* $\mathsf{pk} \leftarrow X_2$.
- $\mathsf{Sign}_{\mathsf{BBS}}(\mathsf{pp}, \mathsf{sk}, (a_1, \ldots, a_m))$. *Compute* $C(\mathbf{a}) = g_1 \prod_{i=1}^{m} h_i^{a_i}$. *Randomly generate* $e \xleftarrow{\$} \mathbb{Z}_p$ *and compute* $A = C(\mathbf{a})^{\frac{1}{e+x}}$. *Output the pair* $(A, e) \in \mathbb{G}_1 \times \mathbb{Z}_p$.
- $\mathsf{Verify}_{\mathsf{BBS}}(\mathsf{pp}, \mathsf{pk}, (A, e), \mathbf{a})$. *Set* $C(\mathbf{a}) = g_1 \prod_{i=1}^{m} h_i^{a_i}$ *and check that* $\mathbf{e}(A, X_2 g_2^e) = \mathbf{e}(C(\mathbf{a}), g_2)$*, or equivalently*

$$\mathbf{e}(A, X_2) = \mathbf{e}(C(\mathbf{a})A^{-e}, g_2). \tag{1}$$

Lemma 1 ([TZ23, Theorem 1]). *The* BBS *signature scheme is strongly unforgeable against chosen messages under the* qSDH *assumption.*

Zero-Knowledge Proofs of Knowledge for BBS Signatures. A few efficient zero-knowledge proofs of knowledge for BBS signatures are given by [TZ23]. We recall for convenience the protocol for Partial Disclosure given in [TZ23, Section 5.2] in Appendix B, Protocol 4.

If we assume that the issuer only issues credentials containing BBS signatures generated according to Definition 4, this protocol is a proof of knowledge of a BBS signature and allows the prover to reveal some of the attributes signed in it. We refer to the set of revealed attributes of the signature with the symbol $\mathsf{Rev} \subseteq [m]$, and to the hidden attribute with the symbol $\mathsf{Hid} = [m] \setminus \mathsf{Rev}$.

At a high level, the prover first randomizes the signature material and then executes a sigma protocol for linear relations. The verifier then checks that the randomized signature material is consistent with the public key of the signer pk, the sigma protocol for linear relations produced a valid response, and that the BBS verification algorithm verifies for the randomized signature material (i.e., $\mathbf{e}(\overline{A}, X_2) = \mathbf{e}(\overline{B}, g_2)$).

Non-interactive and fresh proofs of knowledge. To present in a non-interactive way a BBS credential, a sigma protocol to prove knowledge of the credential (see Figure 2) is made non-interactive by applying the Fiat-Shamir transform. Moreover, in order to be sure that the proof of knowledge of the credential is fresh (i.e. has been created after the session with the verifier has been opened), the verifier sends a random nonce nonce that the prover incorporates into the proof. For completeness, we explicitly describe the presentation algorithm and the verification in Appendix B, Figure 3.

4 Multi-Holder Anonymous Credentials

In this section we introduce the concept of a Multi-Holder Anonymous Credential (MHAC) scheme. At high level, a MHAC scheme allows an issuer to issue shares cred_i of a credential to multiple holders $\mathsf{P}_i, i \in [n]$. Then, if at least a threshold t of the holders agree to present the credential, they can execute a multi-party protocol which returns a valid presentation pres of the credential. However, without the participation of at least t holders, they are unable to produce a valid presentation.

Definition 5 (Multi-holder anonymous credential scheme). *A MHAC scheme consists of the following algorithms:*

- ***Issuer setup algorithm**:*

$$\mathsf{IssSetup}(\kappa) \xrightarrow{\$} (\mathsf{pp}, (\mathsf{pk}, \mathsf{sk})).$$

  This algorithm generates public parameters pp *(e.g. the number of attributes* m*) and the issuer key pair* $(\mathsf{pk}, \mathsf{sk})$*;*
- ***Multi-holder credential issuing protocol**:*

$$\mathsf{CredIss}(\mathsf{pp}, \mathsf{sk}, t, n, \{\mathsf{P}_i\}_{i\in[n]}, \{a_i\}_{i\in[m]}, \mathsf{Prv}) \xrightarrow{\$} \{\mathsf{cred}_i\}_{i\in[n]}.$$

  This protocol is executed by the issuer (possibly interacting with the holders $\mathsf{P}_i, i \in [n]$*) to generate shares* $\{\mathsf{cred}_i\}_{i\in[n]}$ *of a credential with threshold* t *for attributes* $\{a_j\}_{j\in[m]}$*, where the attributes* $\{a_j\}_{j\in\mathsf{Prv}}, \mathsf{Prv} \subseteq [m]$ *are "private" and not necessarily known in the clear to all holders.*
- ***Multi-holder presentation protocol**:*

$$\mathsf{CredPres}(\mathsf{pp}, \mathsf{pk}, t, \{(\mathsf{P}_i, \mathsf{cred}_i)\}_{i\in S}, \{a_i\}_{i\in\mathsf{Rev}}, \mathsf{nonce}) \xrightarrow{\$} \mathsf{pres}.$$

  This protocol is executed by a set $\{\mathsf{P}_i\}_{i\in S}$ *of* t *holders who jointly create a presentation* pres *for* nonce *and public attributes* $\{a_i\}_{i\in\mathsf{Rev}}$*.*
- ***Multi-holder presentation verification algorithm**:*

$$\mathsf{VfPres}(\mathsf{pp}, \mathsf{pk}, \mathsf{nonce}, \{a_i\}_{i\in\mathsf{Rev}}, \mathsf{pres}) \to 0/1.$$

  This algorithm is executed by the verifier who checks if pres *is a valid presentation (for* nonce *and* $\{a_i\}_{i\in\mathsf{Rev}}$*) of a credential* cred *issued by* pk *such that, if* $\{a'_j\}_{j\in[m]}$ *are the attributes included in* cred*, then* $\forall j \in \mathsf{Rev}, a_j = a'_j$*.*

Now we introduce a special class of MHAC scheme which is of practical interest: a MHAC scheme compatible with secure anonymous credential schemes. We say that a MHAC scheme is *compatible* with an anonymous credential scheme if the MHAC is built on top of an existing anonymous credential scheme in a way that:

– an anonymous credential can be reconstructed from t credential shares. Therefore, it is worth defining an algorithm $\mathsf{ReconstructCred}(\{\mathsf{cred}_i\}_{i\in S}), |S| \geq t$ that returns the reconstructed credential cred of the underlying anonymous credential scheme, if the shares $\{\mathsf{cred}_i\}_{i\in S}$ are consistent and valid shares.
– the presentation pres produced by $\mathsf{CredPres}$ has the same structure and is verified in the same way as in the underlying anonymous credential scheme. Moreover, as long as all the holders participating to the presentation protocol are honest, the distribution of the output pres is the same as for the distribution of the presentations of the anonymous credential scheme.

Note that it is straightforward to convert between classic anonymous credentials and their compatible multi-holder variants.

1. To convert a multi-holder version into the single holder, the issuer can simply send t shares to a single party execute the algorithm $\mathsf{ReconstructCred}$ to generate the associated credential and generate the presentation on its own.
2. To convert from a single holder credential to a multi-holder credential, the party holding the full credential acts as the issuer and uses the issuing algorithm to split the credential into shares. It distributes the shares to the other holders and keeps only the share it generated for itself (i.e., it deletes the full credential). In this case, it is desirable that the secret-sharing specified by the MHAC scheme does not rely on specific restrictions on the structure of the underlying single holder credential that, in some cases, might not be satisfied.
   For example, if the secret sharing is performed by distributing an attribute s that is always kept hidden, then it will not be possible for a holder to distribute over multiple devices a credential that is not provided of this extra attribute.

Remark 1. In the above definition, we describe an issue algorithm that outputs credential shares based on credential attributes it takes as input. However, an issuer may be adversarial and the user might want to ensure that the adversary does not learn anything about the private attributes being certified (even while ensuring that these attributes satisfy a particular policy). Thus, as part of our construction, we give a protocol that securely implements the issue algorithm in a way that ensures the security of the private attributes.

5 Security Definitions

In this section we define the security notions associated to MHAC schemes, namely correctness (Section 5.1), unlinkability (Section 5.2), presentation with identifiable abort (Section 5.3), and concurrent unforgeability of presentations (Section 5.4).

Definition 6 (Secure MHAC scheme). *We say that a MHAC scheme is secure if it satisfies the notions of correctness (Definition 7), unlinkability (Definitions 8 and 9), identifiable abort (Definition 11), and concurrent unforgeability of presentations (Definition 12).*

5.1 Correctness

Intuitively, correctness states that running credential presentation with an honestly generated credential will always verify.

Definition 7 (Correctness). *A MHAC scheme is correct if for values* nonce, $\{a_i\}_{i\in[m]}, \mathsf{Rev} \subseteq [m] \setminus \mathsf{Prv}, S \subseteq [n], |S| = t, t \leq n$, *it holds that*

$$1 \leftarrow \mathsf{VfPres}(\mathsf{pp}, \mathsf{pk}, \mathsf{nonce}, \{a_i\}_{i\in\mathsf{Rev}}, \mathsf{pres})$$

where

$$\begin{aligned}(\mathsf{pp}, (\mathsf{pk}, \mathsf{sk})) &\xleftarrow{\$} \mathsf{IssSetup}(\kappa)\\ \{\mathsf{cred}_i\}_{i\in[n]} &\xleftarrow{\$} \mathsf{CredIss}(\mathsf{pp}, \mathsf{sk}, t, n, \{a_i\}_{i\in[m]}, \mathsf{Prv})\\ \mathsf{pres} &\xleftarrow{\$} \mathsf{CredPres}(\mathsf{pp}, \mathsf{pk}, t, \{(\mathsf{P}_i, \mathsf{cred}_i)\}_{i\in S}, \{a_i\}_{i\in\mathsf{Rev}}, \mathsf{nonce})\end{aligned}$$

5.2 Unlinkability

When defining unlinkability, there are two general notions: (1) an adversary cannot "link" usage of the same credential across different presentations and (2) if a credential contains private attributes (i.e., attributes not known to all holders), an adversary cannot learn any information about these private attributes from presentations.

Unlinkability of presentations. This first notion of unlinkability across credential presentations we can only hope to capture in the setting where the credential presentation is generated by *all honest parties*. Intuitively, unlinkability of a credential across different presentations cannot be realized if an adversary participates in the presentation because it inherently must know the credential in order to participate in the protocol. Moreover, to convince another party that a presentation the adversary took part in corresponds to a particular credential, the adversary can reveal the credential and the randomness it used to produce the transcript.

Experiment 1 ($\mathsf{Exp}^{\mathsf{unlink}}_{\mathcal{A}}(\kappa)$ — MHAC Presentation Unlinkability).

1. *The adversary* $\mathcal{A}$ *generates a set of public parameters* pp*, an issuer public key* pk*, and a multi-holder credential* $\{\mathsf{cred}_i\}_{i\in[n]}$ *on attributes* $\{a_i\}_{i\in[m]}$ *of its choosing issued under* pk*. The adversary sends this information to the challenger* $\mathcal{C}$ *together with the information related to the presentation that* $\mathcal{C}$ *must produce, namely* $\mathsf{nonce}, \{a_i\}_{i\in\mathsf{Rev}} \subseteq \{a_i\}_{i\in[m]}$.
2. $\mathcal{C}$ *runs* $\mathsf{pres} \leftarrow \mathsf{CredPres}(\mathsf{pp}, \mathsf{pk}, t, \{(\mathsf{P}_i, \mathsf{cred}_i)_{i\in S}, \{a_i\}_{i\in\mathsf{Rev}}, \mathsf{nonce}\})$ *with a set* $S \subseteq [n]$, $|S| = t$ *and records the transcript of the protocol execution as* T*.* $\mathcal{C}$ *then checks that* $\mathsf{VfPres}(\mathsf{pp}, \mathsf{pk}, \mathsf{nonce}, \{a_i\}_{i\in\mathsf{Rev}}, \mathsf{pres}) \rightarrow 1$ *. If the presentation does not verify,* $\mathcal{C}$ *aborts and the experiment outputs a random bit*

b[9]. *Otherwise,* $\mathcal{C}$ *samples uniformly at random a bit* b. *If* $b = 1$, $\mathcal{C}$ *overwrites* (pres, T) *with the output from a simulated presentation as* $(\mathsf{pres}, T) \leftarrow \mathsf{SimCredPres}(\mathsf{pp}, \mathsf{pk}, t, \tau, \{a_i\}_{i \in \mathsf{Rev}}, \mathsf{nonce})$. *Otherwise,* $\mathcal{C}$ *keeps* (pres, T) *as is.*
3. $\mathcal{C}$ *sends* (pres, T) *to the adversary* $\mathcal{A}$.
4. *If* $b = b'$, *the experiments outputs* 1. *Otherwise the experiment outputs* 0.

Definition 8 (Unlinkability of MHAC presentations). *We say that the presentations of a MHAC scheme are unlinkable if there exist an algorithm* $\mathsf{SimCredPres}(\mathsf{pp}, \mathsf{pk}, t, \{a_i\}_{i \in \mathsf{Rev}}, \mathsf{nonce})$ *such that an adversary* $\mathcal{A}$ *can win* $\mathsf{Exp}^{\mathsf{unlink}}_{\mathcal{A}}(\kappa)$ *with at most negligible advantage. That is,* $\left|\Pr\left[\mathsf{Exp}^{\mathsf{unlink}}_{\mathcal{A}}(\kappa) = 1\right] - \frac{1}{2}\right| \leq \nu(\kappa)$, *where* $\nu(\kappa)$ *is negligible in* κ.

Unlinkability of private attributes. For settings in which some attributes are not known to all holders, we introduce another notion of unlinkability to capture that an adversary does not learn anything about these *secret* attributes when less than t holders are corrupt. Note that these private attributes are determined when the credential is issued and are always a subset of the attributes that are hidden from the verifier.

Experiment 2 ($\mathsf{Exp}^{\mathsf{unlink-attr}}_{\mathcal{A}}(\kappa)$ — MHAC Unlinkability of Private Attributes).

1. *The challenger* $\mathcal{C}$ *runs* $(\mathsf{pp}, (\mathsf{pk}, \mathsf{sk})) \xleftarrow{\$} \mathsf{IssSetup}(\kappa)$ *and sends* $(\mathsf{pp}, (\mathsf{pk}, \mathsf{sk}))$ *to the adversary* $\mathcal{A}$.
2. $\mathcal{A}$ *chooses and sends to* $\mathcal{C}$*:*
   - *a set of attributes* $a_1, \ldots, a_{m-2}$;
   - *two challenge private attributes* $a^{(0)}_{m-1}, a^{(1)}_{m-1}$;
   - *A subset* $\mathsf{cor} \subseteq [n]$ *of parties to corrupt, with* $|\mathsf{cor}| < t$.
3. $\mathcal{C}$ *flips a coin* $b \xleftarrow{\$} \{0, 1\}$ *and runs* $\mathsf{CredIss}$ *with* $\mathcal{A}$ *honestly with private attribute* $a^{(b)}_{m-1}$ *and public attributes* $a_1, \ldots, a_{m-2}$[10]. $\mathcal{C}$ *plays the role of the issuer and of the honest parties, and* $\mathcal{C}$ *sends to* $\mathcal{A}$ *its shares of credential* $\{\mathsf{cred}_i\}_{i \in \mathsf{cor}}$.
4. *The adversary may choose to run* $\mathsf{CredPres}$ *a polynomial number of times with sets* $S \subseteq [n]$ *of size* t, *distinct values* nonce, *and sets* Rev *of its choosing (which do not contain the private attribute), with* $\mathcal{C}$ *playing the role of the honest parties.*
5. *At the end,* $\mathcal{A}$ *sends* $\mathcal{C}$ *its guess* b'. *If* $b = b'$, *the experiment outputs* 1, *otherwise the experiment outputs* 0.

[9] When the MHAC scheme is compatible with an anonymous credential scheme (which is our main case of study), this step can be replaced by an instruction to the challenger to verify the validity of the shares it is provided by executing $\mathsf{ReconstructCred}(\{\mathsf{cred}_i\}_{i \in [n]}) \rightarrow \mathsf{cred}$ and verifying the validity of cred.

[10] Note that the challenger $\mathcal{C}$, which in this experiment acts as an issuer, knows the value of the private attribute. This is not always true in general, in fact a private attribute might be unknown both to the holders and to the issuer.

Definition 9 (Unlinkability of Private Attributes). *We say that the private attributes of a MHAC scheme are unlinkable if any PPT adversary $\mathcal{A}$ can win $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{unlink-attr}}(\kappa)$ with at most negligible advantage. That is,* $\left|\Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{unlink-attr}}(\kappa) = 1\right] - \frac{1}{2}\right| \leq \nu(\kappa)$*, where $\nu(\kappa)$ is negligible in κ.*

Remark 2. Note that Experiment 1 and 2 could be modified to allow the challenger to generate the public parameters using a trapdoor and send them, with the trapdoor, to the adversary. However, our definition, which instructs the challenger to generate the parameters honestly (Experiment 2), or allows the adversary to choose them (Experiment 1), is stronger and encompasses its variant that involves using trapdoors in the parameter generation.

Remark 3. One might ask what the motivation is behind having private attributes not known to holders. The private attributes might be attributes that are sometimes revealed, but only in extremely rare circumstances. In these circumstances, the holders reconstruct the private attribute, reveal it, prove its correctness, and then erase it. For private attributes that are never revealed, we can consider a multi-authority scenario in which issuers use a private attribute to ensure that multiple credentials are issued to the same entity. For example, one issuer may be responsible for physically making sure that a user's holder devices meet an appropriate measure of hardware security; a private attribute can be created by these devices at the time this "device binding" credential is issued. Another issuer can incorporate the secret device binding attribute into the credential it issues to take advantage of the hardware security guarantees that comes with it: the holders can only successfully prove knowledge of this attribute if they are using the appropriate hardware. In this case, the value of the attribute can never be reconstructed. This use is specific to multi-authority case which we do not formally address in this paper, however.

5.3 Presentation with identifiable abort

We adapt the notion of identifiable abort [IOZ14] to credential presentations. Intuitively, in our setting, we wish to capture that a protocol satisfying identifiable abort allows the protocol participants to detect malicious behavior by other participants which would prevent the creation of a valid presentation. Our definition is weaker than what is typically used in general multiparty computation [IOZ14, Appendix B] because we do not aim to realize *any* functionality. We only want to be assured that the protocol does not abort if all the participant are honest, and that when the protocol aborts, at least a corrupted participant is detected.

We do that by defining the notion of an *identifiable abort detector* algorithm which is an algorithm run by each participant P_i of a multi-party protocol Π and allows to determine if one of the other participants P_j has misbehaved.

Definition 10 (Identifiable Abort Detector Algorithm). *Let Π be the multi-party protocol and let $\{\mathsf{P}_i\}_{i \in S}$ be the set of participants to a protocol execution. An* identifiable abort detector *W is an algorithm which can be executed*

by any party $\mathsf{P}_i, i \in S$ *and is a "wrapper" interactive algorithm that relays messages between* Π *algorithm run by* P_i *and the other participants in* S*. Essentially, instead of executing protocol* Π*, party* P_i *executes* $W \circ \Pi$ *which is defined as follows:*

- W *is initialized by the public inputs to the protocol, and it keeps state (on a special state tape) after processing each message exchanged between* Π *and* P_i*.*
- *Each time* Π *sends a message* m *to* P_i*,* m *is forwarded to* W*'s input tape before reaching* P_i*.* W*'s processing of* m *results in either:*
  1. *forwarding* m *to* P_i*: more precisely,* W *clears its input tape, updates its state tape, and outputs* $(\mathsf{message}, m)$ *on its output tape, resulting in the message going through. In this case* P_i *keeps executing* Π*;*
  2. *aborting the protocol and identifying another participant,* P_j*, that deviated from the prescribed protocol: more precisely,* W *writes* (abort, j) *for some* $j \in S$ *on its output tape. In this case* P_i *aborts the protocol and a message* (abort, j) *is broadcast to* Π [11]*.*

Put another way, W observes the incoming communication of a party P_i and has the option to either let the communication through, or to abort the protocol; each time it chooses to abort, it also accuses another participant, j, of maliciously deviating from the protocol. Whenever W outputs (abort, j) for some $j \in S$, it causes P_i to abort the protocol as well.

Definition 11 (Presentation with identifiable abort). *Let* $\mathsf{CredPres}'$ *be a multi-holder credential presentation algorithm. If there exists an efficient identifiable abort detector algorithm* W *for* $\mathsf{CredPres}'$ *such that the following properties hold for the composed algorithm* $\mathsf{CredPres} = W \circ \mathsf{CredPres}'$*:*

- Correctness*: Whenever* $\mathsf{CredPres}$ *never instructs a party to abort, the output* pres *of* $\mathsf{CredPres}$ *verifies, i.e.* $\mathsf{VfPres}(\mathsf{pp}, \mathsf{pk}, \mathsf{nonce}, \{a_i\}_{i \in \mathsf{Rev}}, \mathsf{pres}) = 1$*.*
- Identifiability*: Whenever* $\mathsf{CredPres}$ *outputs a message is* (abort, j) *for some* $j \in S$*,* P_j *did not follow the protocol instructions and is therefore corrupt.*

We say that $\mathsf{CredPres}$ *satisfies identifiable abort.*

Remark 4. Note that this property is not concerned with assuring that only a legitimate holder can carry out the presentation protocol without being detected. Legitimacy of the credential being presented is addressed in the unforgeability of presentations property described in Section 5.4. For identifiable abort, we only want to be assured that if a holder would cause the protocol to output an invalid presentation, the honest parties can identify the this holder. Conversely, if the algorithm does not abort, the presentation will be valid.

[11] This instance covers the case where another participant has output a message (abort, k).

5.4 Concurrent unforgeability of presentations

We describe an experiment defining the unforgeability of a multi-holder anonymous credential presentation algorithm $\mathsf{CredPres}$. The experiment resembles the security experiment for threshold signature schemes. We can think of the shares of the t-of-n multi-holder credential as shares of the signing key in a threshold signature scheme. The message that gets signed is the nonce nonce provided by the verifier before the presentation is created (see Figure 3).

If the adversary has t or more shares of a t-of-n multi-holder anonymous credential, we will write that the adversary is given a "full credential", since with t shares the adversary can produce presentations on its own.

The experiment is divided in three phases: a *Setup phase*, a *Training phase* and a *Forgery phase*. In the Setup phase, the challenger generates the parameters and credential issuing keys. During the Training phase, the forger is allowed polynomially many queries to an issuing oracle and a credential presentation oracle. There are two types of issuing queries: (1) a query for a full credential where the adversary gets all the shares of the credential and can present it on its own from now on; and (2) a query for a "target" credential for which the adversary is only provided a subset of fewer than t shares. We limit the adversary to just one such target query; this is without loss of generality (see Observation 4 below).

Finally, in the Forgery phase, the forger outputs a tuple consisting of a nonce, attributes, and credential presentation. If this tuple verifies and the contents of the tuple do not correspond to a credential produced by the issuing oracle or a presentation output by the presentation oracle, then the forger wins the experiment. Otherwise, the forger loses.

The experiment $\mathsf{Exp}^{\mathsf{c-uf-pres}}_{\mathcal{A}}$ is given below and summarized in Figure 5.

Experiment 3 ($\mathsf{Exp}^{\mathsf{c-uf-pres}}_{\mathcal{F}}$ — Concurrent unforgeability of MHAC presentation).

Setup phase. The challenger executes $\mathsf{IssSetup}(\kappa)$*, which returns the set of public parameters* pp *and a key pair* $(\mathsf{sk}, \mathsf{pk})$*. The challenger sends* $(\mathsf{pp}, \mathsf{pk})$ *to* $\mathcal{F}$*.*

Training phase. The forger $\mathcal{F}$ *has access to two oracles,* $\mathcal{O}_{\mathsf{iss}}$ *and* $\mathcal{O}_{\mathsf{pres}}$*, which it may query in the following ways:*

- $\mathcal{F}$ *can query an issuing oracle* $\mathcal{O}_{\mathsf{iss}}$ *for a polynomial number* q_I *of full credentials* cred *for attributes* $\{a_i\}_{i\in[m]}$ *of its choice, and one single query for a partial credential (target credential)* $\{\mathsf{cred}_i\}_{i\in\mathsf{cor}}$ *for* $\{a_i\}_{i\in[m]}$*.*[12]
  - Issuance of full credentials*: on input the set of attributes* $\{a_i\}_{i\in[m]}$ *chosen by* $\mathcal{F}$*,* $\mathcal{O}_{\mathsf{iss}}$ *provides* $\mathcal{F}$ *with a credential* $\mathsf{cred} = \{\mathsf{cred}_i\}_{i\in[n]} \xleftarrow{\$} \mathsf{CredIss}(\mathsf{pp}, \mathsf{sk}, n, t, \{a_i\}_{i\in[m]}, \mathsf{Prv})$ *on these attributes.*
    $\mathcal{O}_{\mathsf{iss}}$ *stores a record* (cred) *in a credential table* CT*.*

[12] In this experiment we always assume that the adversary knows all the attributes included in the credentials it is issued, therefore we do not need to mention the private attributes.

- Issuance of the target credential*: $\mathcal{F}$ gives as input to $\mathcal{O}_{\mathsf{iss}}$ the tuple $(\{a_i\}_{i\in[m]}, t, n, \mathsf{cor})$ where $\{a_i\}_{i\in[m]}$ are attributes chosen by $\mathcal{F}$ to include in the credential, (t, n) are the parameters of the secret sharing of the credential, and $\mathsf{cor} \subset [n], |\mathsf{cor}| < t$, are the parties $\mathcal{F}$ wants to corrupt.*
  $\mathcal{O}_{\mathsf{iss}}$ computes $\mathsf{CredIss}(\mathsf{pp}, \mathsf{sk}, t, n, \{a_i\}_{i\in[m]}, \mathsf{Prv}) \to \{\mathsf{cred}_i\}_{i\in[n]}$ and gives to $\mathcal{F}$ only the shares corresponding to the parties in cor.
  $\mathcal{O}_{\mathsf{iss}}$ stores the value $\mathsf{targetCred} \leftarrow (\{\mathsf{cred}_i\}_{i\in[n]}, n, t, \mathsf{cor})$.

– *$\mathcal{F}$ can query a presentation oracle $\mathcal{O}_{\mathsf{pres}}$ for a polynomial number q_P of presentations of the target credential specifying the nonce nonce to use and the attributes $\{a_i\}_{i\in\mathsf{Rev}}$ to reveal [13]. We allow the adversary to open concurrently many sessions of the presentation protocol for the target credential and interleave messages between different sessions. Therefore, to distinguish sessions, $\mathcal{F}$ includes a unique session identifier ssid to messages sent to $\mathcal{O}_{\mathsf{pres}}$. To simplify the description, we will omit ssid which is included in every message exchanged between the holders.*

- Presentation of the target credential $\mathsf{targetCred}$*: $\mathcal{F}$ gives in input to $\mathcal{O}_{\mathsf{pres}}$ the tuple $(\mathsf{nonce}, \{a_i\}_{i\in\mathsf{Rev}}, \mathsf{hon})$ which specifies the nonce for the presentation, the set of attributes to reveal and the set of parties $\mathsf{hon} \subseteq [n] \setminus \mathsf{cor}$ s.t. $|\mathsf{cor}| + |\mathsf{hon}| = t$.*
  $\mathcal{O}_{\mathsf{pres}}$, controlling hon, interacts with $\mathcal{F}$, controlling cor, in the execution of $\mathsf{CredPres}(\{\mathsf{P}_i, \mathsf{cred}_i\}_{i\in\mathsf{cor}\cup\mathsf{hon}}, \{a_i\}_{i\in\mathsf{Rev}}, \mathsf{nonce}, \mathsf{pp}, \mathsf{pk})$. If $\mathcal{O}_{\mathsf{pres}}$ sends its last protocol message associated to that specific session, it stores in the presentation table PT the record $(\mathsf{nonce}, \{a_i\}_{i\in\mathsf{Rev}})$ [14].

Forgery phase. At the end of the training, $\mathcal{F}$ produces a forgery $(\mathsf{nonce}^\star, \{a_i^\star\}_{i\in\mathsf{Rev}^\star}, \mathsf{pres}^\star)$ given by a presentation $\mathsf{pres}^\star$ for $(\mathsf{nonce}^\star, \{a_i^\star\}_{i\in\mathsf{Rev}^\star})$ of its choice.

$\mathcal{F}$ wins the experiment if $\mathsf{VfPres}(\mathsf{nonce}^\star, \{a_i^\star\}_{i\in\mathsf{Rev}}, \mathsf{pres}^\star) = 1$ and the following win conditions *related to the queries made by $\mathcal{F}$ are satisfied:*

– *For every record $(\mathsf{nonce}, \{a_i\}_{i\in\mathsf{Rev}})$ in the presentation table PT:*
  $(\mathsf{nonce}^\star, \{a_i^\star\}_{i\in\mathsf{Rev}^\star}) \neq (\mathsf{nonce}, \{a_i\}_{i\in\mathsf{Rev}})$.
  This check guarantees that the forgery is not a forgery generated in a presentation query of the target credential.
– *For every record cred in CT, being $\{a_i\}_{i\in[m]}$ the attributes associated to cred, $\{a_i\}_{i\in\mathsf{Rev}^\star} \neq \{a_i^\star\}_{i\in\mathsf{Rev}^\star}$.*
  This guarantees that the forgery is not derived from a full credential that has been issued by $\mathcal{O}_{\mathsf{iss}}$.

Observation 1. *In this security game, we consider the issuance of credentials as an algorithm which is executed by the issuer given the adversary's input*

[13] Note that $\mathcal{F}$ can generate presentations for the full credentials on its own, without the help of any oracle, and since it can query for the issuance of full credentials, we omit the ability to query presentations of credentials it does not control.

[14] $\mathcal{O}_{\mathsf{pres}}$ does not store the presentation output of the protocol execution because it might not learn its value since in the protocol execution it always sends its messages first

$(\{a_i\}_{i\in[m]}, t, n, \mathsf{cor})$. *However, in general, the issuing of credentials might happen via an issuing protocol which* allows *an adversary to keep some attributes hidden from the issuer, so we should allow the adversary to make queries for credentials without sending all the attributes in the clear as we do. However, this kind of query can be omitted in the security definition if we require issuing protocols that always allow the challenger of the experiment (acting on behalf of the issuer) to extract the attribute values, even when the adversary tries to keep them hidden, for example by means of straight-line extractable NIZKPs.*

Definition 12 (Concurrent unforgeability of MHAC presentations). *We say that a MHAC scheme has concurrently unforgeable presentations if for any PPT adversary $\mathcal{F}$, $\mathcal{F}$ wins with at most negligible probability in $\mathsf{Exp}_{\mathcal{F}}^{\mathsf{c-uf-pres}}(\kappa)$. That is,* $\Pr\left[\mathsf{Exp}_{\mathcal{F}}^{\mathsf{c-uf-pres}}(\kappa) = 1\right] \leq \nu(\kappa)$, *where $\nu(\kappa)$ is negligible in κ.*

Observation 2. *In practical scenarios, the nonce is sent to the provers by a verifier who wants to receive a fresh presentation (see Figure 3). Therefore, if a presentation protocol is unforgeable, i.e. the adversary can not forge a presentation for attributes $\{a_i\}_{i\in\mathsf{Rev}}$ and a nonce* nonce *of its choice, then it will not succeed in forging a presentation for a nonce chosen by the verifier.*

Observation 3. *We remark that our unforgeability experiment (Experiment 3) also captures the standard unforgeability for anonymous credentials. In our definition, an adversary can win Experiment 3 by either producing a presentation forgery of the target credential or by producing a presentation for a (full) credential that was never queried by the adversary. An adversary that forges credentials in the traditional sense wins the unforgeability experiment via the latter condition.*

Observation 4. *Note that we could allow the adversary of the unforgeability game to receive a polynomial number $q_{Ip} = q_{Ip}(\kappa)$ of partial credentials. It is easy to see that a scheme secure according to our definition of security is secure also according to this stronger notion of security. However, the reduction to the cryptographic assumption would reduce its tightness by a factor $\frac{1}{q_{Ip}}$, which is non-negligible in κ. This would impact the dimension of the parameters when it comes the time to instantiate the scheme.*

6 BBS Multi-Holder Anonymous Credentials

In this section we describe a secure MHAC scheme which is compatible with the BBS anonymous credential scheme [TZ23]. According to the definition of MHAC scheme compatible with an anonymous credential scheme, the credential issuance algorithm consists in computing a secret sharing of a BBS credential, and the presentation structure is the same as the one presented by Tessaro and Zhu in [TZ23, Section 5] (Figure 3) .

Design principle. Every issuer can decide the structure, or schema, of the credentials it issues, determining, for example, (1) the number of attributes, which could even be zero, (2) the semantic meaning of the attributes and (3) the possible values associated with each attribute, ranging from the binary value to all $\mathbb{Z}_p$. As we have mentioned in Section 4, it is desirable to design a MHAC scheme compatible with an anonymous credential scheme that does not require a specific structure of the underlying anonymous credential. This, to take full advantage of the compatibility of the MHAC scheme and to consistently ensure that a holder can convert any credential it is provided into a multi-holder credential. The only way to achieve this, and to have a secret sharing completely independent of the credential structure, is to secret share the signature component, which in this work is done by distributing the value e of the BBS signature (A, e).

Private attributes. Our construction (optionally) allows private attributes; they are secret-shared by the holders. Attributes not known in the clear are denoted by the set Prv, and attributes known by all holders are denoted as Pub. Though our protocols are described in terms of t-of-n Shamir secret sharing, replacing the sharing algorithm enables using different access structures (e.g., enforcing that one party always participates in presentations). This extension is given in Section 6.4.

6.1 Credential issuing

In this section, we describe protocols involving the issuer. The issuer setup (Algorithm 1) only needs to be run once locally by the issuer.

Algorithm 1 (Issuer setup algorithm).

$$\mathsf{IssSetup}_{\mathsf{BBS}}(\kappa) \xrightarrow{\$} (\mathsf{pp}, (\mathsf{pk}, \mathsf{sk}))$$

The algorithm $\mathsf{IssSetup}_{\mathsf{BBS}}(\kappa)$ *works as follows.*

1. $\mathsf{Pgen}_{\mathsf{BBS}}(\kappa) \to \mathsf{pp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2, h_1, \ldots, h_m)$
2. $\mathsf{KGen}_{\mathsf{BBS}}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{pk}) = (x, g_2^x)$
3. *Output* $(\mathsf{pp}, (\mathsf{pk}, \mathsf{sk}))$

The credential issuance protocol can be run by the issuer with any set of n holders. We give two variants of credential issuance $\mathsf{CredIss}_{\mathsf{BBS}}$: one for issuing a credential when there are no private attributes (Protocol 1) and another when there are private attributes (Protocol 2).

Credential Issuance without Private Attributes. In the case where all attributes are known in the clear, the holders simply supply the attributes to the issuer, and the issuer can produce the shares of the credential locally. Upon receiving attributes $\{a_i\}_{i\in[m]}$, the issuer creates a credential as follows.

Protocol 1. $\mathsf{CredIss}_{\mathsf{BBS}}(\mathsf{pp}, \mathsf{sk}, \{a_i\}_{i\in[m]})$ — Multi-holder issuing protocol (without private attributes)

1. Compute a BBS signature as $(A, e) \xleftarrow{\$} \mathsf{Sign}_{\mathsf{BBS}}(\mathsf{sk}, \{a_i\}_{i\in[m]})$
2. $\{e^{(i)}\}_{i\in[n]} \xleftarrow{\$} \mathsf{Share}(t, n, e)$
3. For $i \in [n]$, set $\mathsf{cred}_i = (A, e^{(i)}, \{D_j\}_{j\in[n]\setminus\{i\}}, (\{a_j\}_{j\in[m]}, \perp))$ with $D_j = A^{-e^{(j)}}$ and output cred_i to party P_i.

Credential Issuance with Private Attributes. In the case where some attributes may not be known to all holders, each party's credential will have a share of each private attribute rather than the full attribute itself. Let Prv denote the set of private attributes and Pub the set of attributes known by each holder.

Our starting point here is multi-base Pedersen verifiable secret sharing (VSS) [Ped91][15], for example as presented by Cachin et al. [CKLS02] (but with threshold $t < n/2$ since we are in the synchronous case). That is, for each private attribute $a_j \in \mathsf{Prv}$, each party P_i's share of the credential contains a Shamir secret share $a_j^{(i)}$; additionally, the m^{th} attribute a_m is always a private attribute that is meant to serve as the randomness for Pedersen VSS, so P_i also has a Shamir share of it, $a_m^{(i)}$.

To simplify our notation, we will include m in the set of private attributes Prv and $[m] = \mathsf{Prv} \cup \mathsf{Pub}$.

Finally, for each P_i, a share of multi-base Pedersen commitment $C_i = \prod_{j\in\mathsf{Prv}} h_j^{a_j^{(i)}}$ to these attribute shares is known. We assume this was set up prior to the protocol's execution and that each holder has also published a straight-line extractable [Fis05,KS22,LR22,CDG+24] proof of knowledge π_i of these secret shares.

To create a credential with private attributes, the issuer performs the following:

Protocol 2. $\mathsf{CredIss}_{\mathsf{BBS}}(\mathsf{pp}, \mathsf{sk}, \{\pi_i\}_{i\in[n]}, \{C_i\}_{i\in[n]}, \mathsf{Prv}, \{a_i\}_{i\in\mathsf{Pub}})$ — Multi-holder issuing protocol (with private attributes)

1. For each P_i, verify proof π_i corresponding to each C_i, and verify that $\{C_i\}_{i\in[n]}$ are consistent with a Pedersen VSS of $C = \prod_{j\in\mathsf{Prv}} h_j^{a_j}$.
2. Compute $C(\mathbf{a}) = g_1 C \prod_{j\in\mathsf{Pub}} h_j^{a_j}$. Pick a random e and compute $A = C(\mathbf{a})^{1/(x+e)}$.

[15] The private attributes may not be known by the holders and may not be known even by the issuer. If the holders do not know the private attribute, the Pedersen VSS can be executed starting from a value known by the issuer who divides it in shares, or by the holders who generate the secret sharing of an unknown attribute [Ped91, Section 5.2], and in this case not even the issuer will know this value.

3. Generate a secret sharing of e, $\{e^{(i)}\}_{i\in[n]} \xleftarrow{\$} \mathsf{Share}(t, n, e)$.
4. For all $k \in [n]$, compute

$$D_k = C_k A^{-e^{(k)}} = \prod_{j\in\mathsf{Prv}} h_j^{a_j^{(k)}} A^{-e^{(k)}},$$

then set, for all $i \in [n]$

$$\mathsf{cred}_i = (A, e^{(i)}, \{D_k\}_{k\in[n]\setminus\{i\}}, (\{a_j\}_{j\in\mathsf{Pub}}, \{a_j^{(i)}\}_{j\in\mathsf{Prv}})),$$

and output cred_i to party P_i.

Observation 5. *Note that, in case one of the private attributes is never revealed and it is secret-shared using a t-out-of-n Shamir secret sharing, it is not necessary to secret-share also the value e. In that case, the values $D_i = \prod_{j\in\mathsf{Prv}} h_j^{a_j^{(k)}}$ and $\mathsf{cred}_i = (A, e, \{D_k\}_{k\in[n]\setminus\{i\}}, (\{a_j\}_{j\in\mathsf{Pub}}, \{a_j^{(i)}\}_{j\in\mathsf{Prv}}))$.*

However, to keep the presentation of the scheme consistent with the case where private attributes (1) are not used, or (2) are distributed in a way different from the $(t, n)-$Shamir secret sharing, or (3) might be revealed in rare circumstances (see Remark 3), in our description of the protocol we secret-share also the value e.

6.2 Multi-holder presentation

An overview of the presentation protocol is depicted in Appendix C, Figure 4. We recall that the attributes revealed, denoted as Rev, is a subset of the public attributes Pub. The remaining attributes *not* revealed to the verifier are denoted as Hid. An extension for handling attributes shared only among a subset of $n' < n$ holders is described in Section 6.4.

Every presentation protocol execution is associated with a unique session identifier ssid which is included in every message sent by the participants over the private broadcast channel, therefore, we will omit it in our description.

This protocol is run by a subset $\{\mathsf{P}_i\}_{i\in S} \subseteq \{\mathsf{P}_1, \ldots, \mathsf{P}_n\}, |S| = t$, with each party $\mathsf{P}_i \in S$ holding a share of a credential

$$\mathsf{cred}_i = (A, e^{(i)}, \{D_j\}_{j\in[n]\setminus\{i\}}, (\{a_j\}_{j\in\mathsf{Pub}}, \{a_j^{(i)}\}_{j\in\mathsf{Prv}})).$$

Protocol 3. $\mathsf{CredPres}_{\mathsf{BBS}}$ — Multi-holder presentation protocol

Let $j \in S$ refer to a designated "primary party" P_j. Upon receiving the nonce

nonce from a verifier to present a credential with a set of revealed attributes $\mathbf{a}' = \{a_j\}_{j \in \mathsf{Rev}}$, parties P_i for $i \in S$ produce the credential presentation as follows.

Signature material randomization phase. Parties begin the presentation by first producing randomness.

1. The primary P_j first samples an element $r \xleftarrow{\$} \mathbb{Z}_p$ broadcasts r to every other party in S.
2. Every party P_i for $i \in S$ computes:

$$\overline{A} = A^r, \quad D = \prod_{k \in S} D_k{}^{\lambda_{S,k}(0)}, \quad C(\mathbf{a}') = g_1 \prod_{j \in \mathsf{Rev}} h_j^{a_j},$$
$$\tilde{B}_j = \Big(C(\mathbf{a}') \cdot \big(\prod_{k \in \mathsf{Hid} \setminus \mathsf{Prv}} h_k^{a_k}\big) \cdot D_j^{\lambda_{S,j}(0)}\Big)^r, \quad \tilde{B}_i = \Big(D_i^{\lambda_{S,i}(0)}\Big)^r,$$
$$\overline{B} = \prod_{i \in S} \tilde{B}_i = \Big(C(\mathbf{a}') \cdot \big(\prod_{k \in \mathsf{Hid} \setminus \mathsf{Prv}} h_k^{a_k}\big) \cdot D\Big)^r.$$

   where $\lambda_{S,i}(0)$ denotes the Lagrange coefficient for interpolating party P_i's share with the parties indexed by S. Actually, $\overline{B}$ can be computed only by the primary party.

Sigma protocol execution phase. The participants next jointly generate a proof of knowledge of a representation of $\overline{B}$ w.r.t. $C(\mathbf{a}'), \{h_i\}_{i \in \mathsf{Hid}}, \overline{A}$.

3. Parties begin the proof by doing the following:
   - P_j samples $\alpha^{(j)}, \{\beta_i^{(j)}\}_{i \in \mathsf{Hid}}, \gamma^{(j)} \xleftarrow{\$} \mathbb{Z}_p$ and computes
   $$U_j = C(\mathbf{a}')^{\alpha^{(j)}} \cdot \prod_{i \in \mathsf{Hid}} h_i^{\beta_i^{(j)}} \cdot \overline{A}^{\gamma^{(j)}}$$
   .
   - Every other party P_k for $k \in S \setminus \{j\}$ instead samples $\{\beta_i^{(k)}\}_{i \in \mathsf{Prv}}, \gamma^{(k)} \xleftarrow{\$} \mathbb{Z}_p$ and computes
   $$U_k = \prod_{i \in \mathsf{Prv}} h_i^{\beta_i^{(k)}} \cdot \overline{A}^{\gamma^{(k)}}$$
   .

   All the participants P_i, for $i \in S$, then compute commitments to their U_i as $\mathsf{com}_i = \mathcal{H}_{\mathsf{com}}(\mathsf{ssid}, \mathsf{nonce}, U_i)$ and broadcast com_i to the other parties.
4. Upon receiving com_k from every other party $k \in S \setminus \{i\}$, each P_i opens its commitment by broadcasting U_i to every other party.
5. For each U_k that party P_i receives from each P_k, for $k \in S \setminus \{i\}$, if U_k is not a valid opening for com_k, then P_i outputs (abort, k) and aborts.

6. For each $k \in S$, P_k computes:

$$U = \prod_{i \in S} U_i, \quad \mathsf{ch} = \mathcal{H}_{\mathsf{sig}}\left(\mathsf{ssid}, \mathsf{nonce}, U, \overline{A}, \overline{B}, \{a_i\}_{i \in \mathsf{Rev}}\right),$$
$$\left\{z_i^{(k)}\right\}_{i \in \mathsf{Prv}} = \left\{\beta_i^{(k)} + \mathsf{ch}\left(r \cdot a_i^{(k)} \cdot \lambda_{S,k}(0)\right)\right\}_{i \in \mathsf{Prv}},$$
$$z_e^{(k)} = \gamma^{(k)} + \mathsf{ch}\left(-e^{(k)} \cdot \lambda_{S,k}(0)\right)$$

and broadcasts $\{z_i^{(k)}\}_{i \in \mathsf{Prv}}$ and $z_e^{(i)}$.
The primary P_j additionally computes and broadcast

$$z_r^{(j)} = \alpha_j + \mathsf{ch} \cdot r, \quad \left\{z_i^{(j)}\right\}_{i \in \mathsf{Hid} \setminus \mathsf{Prv}} = \left\{\beta_i^{(j)} + \mathsf{ch} \cdot (a_i r)\right\}_{i \in \mathsf{Hid} \setminus \mathsf{Prv}}$$

and broadcasts $z_r^{(j)}, \{z_i^{(j)}\}_{i \in \mathsf{Hid} \setminus \mathsf{Prv}}$.

7. Upon P_i receiving $\{z_i\}_{i \in \mathsf{Hid}}, z_e^{(j)}, z_r^{(j)}$ from the primary P_j, check

$$U_j \cdot \tilde{B}_j^{\mathsf{ch}} \stackrel{?}{=} C(\mathbf{a}')^{z_r^{(j)}} \cdot \prod_{i \in \mathsf{Hid}} h_i^{z_i^{(j)}} \cdot \overline{A}^{z_e^{(j)}}.$$

If the equality does not hold, then P_i outputs (abort, j) and aborts. Otherwise, upon receiving $z_e^{(k)}, \{z_i^{(k)}\}_{i \in \mathsf{Prv}}$ from party P_k for $k \in S \setminus \{j\}$, check

$$U_k \cdot (\tilde{B}_k)^{\mathsf{ch}} \stackrel{?}{=} \prod_{i \in \mathsf{Prv}} h_i^{z_i^{(k)}} \overline{A}^{z_e^{(k)}}.$$

If the equality does not hold, then P_i outputs (abort, k) and aborts.

8. For each $k \in S$, party P_k computes

$$z_r = z_r^{(j)}, \quad \{z_i\}_{i \in \mathsf{Hid} \setminus \mathsf{Prv}} = \left\{z_i^{(j)}\right\}_{i \in \mathsf{Hid} \setminus \mathsf{Prv}},$$
$$\{z_i\}_{i \in \mathsf{Prv}} = \left\{\sum_{i' \in S} z_i^{(i')}\right\}_{i \in \mathsf{Prv}}, \quad z_e = \sum_{i \in S} z_e^{(i)}.$$

where j corresponds to the index of the primary. P_k sets

$$\mathsf{pres} \leftarrow \left(\overline{A}, \overline{B}, \mathsf{ch}, (z_r, \{z_i\}_{i \in \mathsf{Hid}}, z_e)\right)$$

and outputs the tuple $(\mathsf{nonce}, \mathsf{pres})$ as the output of the protocol.

Note that it is crucial to include the revealed attributes in the challenge computation (Step 6) to avoid the principal revealing the attributes in a subset

of $\mathsf{Hid} \cap \mathsf{Pub}$ different from the one agreed with the other parties participating in the presentation protocol.

When distributing the value e is unnecessary (see Observation 5), the presentation protocol must be modified so that the principal carries out the creation of the response corresponding to $\overline{A}$. This leads to a straightforward variant of the presentation protocol whose security can be proved as an exercise.

Comparing computational and communication cost to BBS We can evaluate the cost of our protocol as a function of the number t of parties participating in the protocol, hidden attributes h in the presentation, of which p are private attributes $p < h$, and the number of attributes m. The principal party performs 4 broadcasts and, omitting the computation of D, computes the following exponentiations:

- in the second step: 1 to compute $\overline{A}$, $(m-h)$ to compute $C(\mathbf{a}')$, $h-p+2$ to compute $\tilde{B}_j$, $t-1$ to compute all the $\tilde{B}_i, i \in S \setminus \{j\}$;
- in the third step: $h+2$ exponentiations to compute U_j;

for a total amount of $m+h-p+t+4$ exponentiations

The other parties each perform only 3 broadcasts and computes the following exponentiations:

- in the second step : 1 to compute $\overline{A}$, $(m-h)$ to compute $C(\mathbf{a}')$, $h-p+2$ to compute $\tilde{B}_j$, $t-1$ to compute all the $\tilde{B}_i, i \in S \setminus \{j\}$;
- in the third step: $p+1$ exponentiations to compute U_k;

for a total amount of $m+t+3$ exponentiations.

Part of these exponentiations are executed to perform the identifiable abort checks; if we omit these checks, the number of exponentiations is reduced because the party P_i does not have to compute the values $\tilde{B}_k$ for $k \neq i$.

The centralized case described in [TZ23] requires the following exponentiations:

- 1 to compute $\overline{A} = A^r$;
- $m-h$ to compute $C(\mathbf{a}')$ and other $h+2$ to compute $\overline{B}$;
- $h+2$ to compute the proof of knowledge of a representation of $\overline{B}$.

For a total number of $m+h+5$ exponentiations.

6.3 Verification

Since our MHAC scheme is compatible with the BBS anonymous credential scheme, the verification algorithm is exactly the same as the one described in [TZ23].

Algorithm 2 (Multi-holder presentation verification algorithm).

$$\mathsf{VfPres}_{\mathsf{BBS}}(\mathsf{pp}, \mathsf{pk}, \mathsf{nonce}, \{a_i\}_{i \in \mathsf{Rev}}, \mathsf{pres}) \to 0/1$$

Let $\mathsf{pres} = \left(\overline{A}, \overline{B}, \mathsf{ch}, (z_r, \{z_i\}_{i\in\mathsf{Hid}}, z_e)\right)$. *The verifier runs the same verification algorithm as in the centralized case [TZ23]:*

$$U \leftarrow \overline{B}^{-\mathsf{ch}} C(\mathbf{a}')^{z_r} \prod_{i\in\mathsf{Hid}} h_i^{z_i} \overline{A}^{z_e},$$

$$\mathsf{ch} \stackrel{?}{=} \mathcal{H}_{\mathsf{sig}}\left(\mathsf{ssid}, \mathsf{nonce}, U, \overline{A}, \overline{B}, \{a_i\}_{i\in\mathsf{Rev}}\right), \quad \mathbf{e}(\overline{A}, X_2) \stackrel{?}{=} \mathbf{e}(\overline{B}, g_2).$$

If the relations hold, the verifier outputs 1. *Otherwise, outputs* 0.

6.4 Extensions

Flexible presentation subsets. Let us refer to any subset of holders who can present a MHAC using their shares of credential as a *presentation subset* for the given credential. In this work we have described a scheme where the attributes $\{a_j\}_{j\in\mathsf{Prv}}$ are shared among the holders in an homogeneous way using a (t, n)-Shamir secret sharing, so any subset of t parties is a presentation subset.

This construction can be easily generalized, allowing the issuer to share one attribute only among a subset of the holders (performing a (t', n')- Shamir secret sharing with $n' < n$), or even to a single holder (in this case, the cooperation of this holder will be necessary to create the presentation). Therefore, the presentation subsets can be any subset of holders that know enough shares for each attribute. The participants will also be required to deterministically choose a factorization of $\overline{B}$ which allows them to generate the proof of knowledge of the representation in a coordinated way.

Share size optimization. In Appendix H we describe an optimization to the size of the shares of the credentials. As currently given, the size of each credential share is linear in the number n of participants due to each party knowing the values D_i of every other group member. This can be reduced by having the issuer give each party P_i only its own value D_i along with a signature σ_i on D_i and some values binding D_i to the multi-holder anonymous credential. In the first step of the presentation protocol, the participants broadcast their values com_i together with the values (D_i, σ_i) corresponding to their share and the issuer's signature.

Distributing the issuer. Note that while our issuing protocol (Protocol 1) is described in terms of a single issuer, distributing the issuer can be achieved by replacing computation of the BBS component (Steps 1 and 2) with a distributed protocol such as [DKL+23].

7 Security Analysis

In this section we prove our BBS MHAC scheme from Section 6 satisfies the security properties defined in Section 5. We split the proof into four parts, showing that our BBS MHAC satisfies correctness, unlinkability, identifiable abort, and unforgeability.

Theorem 3. *Let* $\Pi_{\mathsf{MHAC-BBS}} = (\mathsf{IssSetup}_{\mathsf{BBS}}, \mathsf{CredIss}_{\mathsf{BBS}}, \mathsf{CredPres}_{\mathsf{BBS}}, \mathsf{VfPres}_{\mathsf{BBS}})$. *Assuming* BBS *is SUF-CMA and the* DL *assumption holds in our group* $\mathbb{G}_1$, $\Pi_{\mathsf{MHAC-BBS}}$ *is a concurrently secure MHAC scheme in the programmable random oracle model satisfying the security properties in Section 5 against an active static adversary corrupting less than* t *holders and an honest-but-curious issuer.*

The proof follows from Lemmas 2, 3, 4, 5, and 6.

7.1 Correctness of BBS MHAC

Lemma 2. $\Pi_{\mathsf{MHAC-BBS}}$ *satisfies correctness (Definition 7).*

Proof. Checking the steps of the presentation protocol it is easy to see that the holders in possession of a BBS MHAC execute in a multi-party fashion the same operations described in the presentation protocol in [TZ23], namely they perform the signature material randomization phase computing $\overline{A}$ and $\overline{B}$ such that $\mathbf{e}(\overline{A}, X_2) = \mathbf{e}(\overline{B}, g_2)$, and then they execute the sigma protocol execution phase which outputs a proof of knowledge of $\overline{B}$ w.r.t. $C(\mathbf{a}), \{h_i\}_{i \in \mathsf{Hid}}, \overline{A}$. Therefore the correctness of $\Pi_{\mathsf{MHAC-BBS}}$ follows from the correctness of the algorithm for the presentation of a BBS anonymous credential.

7.2 Unlinkability of presentations of BBS MHAC.

Lemma 3. $\Pi_{\mathsf{MHAC-BBS}}$ *satisfies presentation unlinkability (Definition 8) in the programmable random oracle model.*

Proof. To prove unlinkability, we show there exist an algorithm $\mathsf{SimCredPres}(\cdot)$ which simulates an honest presentation of a multi-holder credential.

Regarding the multi-holder BBS anonymous credential scheme, it being compatible with the BBS anonymous credential scheme [TZ23], we can choose as $\mathsf{SimCredPres}(\mathsf{pp}, \mathsf{pk}, \tau, \{a_i\}_{i \in \mathsf{Rev}}, \mathsf{nonce})$ the same algorithm used to simulate the generation of presentation of a BBS anonymous credential presented in [TZ23] [16] that we recall in Appendix B. The transcript T of the communication between the participants is instead generated as a random string of a given length which is indistinguishable from a real transcript since the participants execute the protocol over a private broadcast channel.

Since the challenger of the experiment programs the random oracle, the simulated presentation is indistinguishable from the real one, and the simulation fails only with negligible probability if we allow the adversary to query the random oracle a polynomial number of times. □

[16] We recall that, together with the public key pk, the adversary must provide the challenger $\mathcal{C}$ with a pair (U_1, U_2) such that $\mathbf{e}(U_1, \mathsf{pk}) = \mathbf{e}(U_2, g_2)$ which the simulator must use to simulate the generation of the values $\overline{A}, \overline{B}$. Such a pair is assumed to be known for every BBS credential issuer because it can be obtained from any presentation of any credential issued by that specific issuer, as it is specified in [TZ23,CDL16,LKWL22].

7.3 Unlinkability of private attributes of BBS MHAC

Lemma 4. *$\Pi_{\mathsf{MHAC-BBS}}$ satisfies private attribute unlinkability (Definition 9).*

We provide a proof sketch below and give the formal proof in Appendix D.

Proof Sketch. It is possible to design a reduction to the hiding property [KL07] of the Pedersen commitment scheme [Ped91] which is perfectly hiding.

The adversary $\mathcal{A}$ of private attribute unlinkability sends to the challenger (i.e. the reduction) $\mathcal{B}$ two attributes a_0^*, a_1^* and the set of public attributes $\{a_i\}_{i \in \mathsf{Pub}}$. The reduction $\mathcal{B}$ sends the same messages to the challenger $\mathcal{C}$ of the hiding property of Pedersen commitment who samples a bit b uniformly at random and computes a commitment $c \in \mathbb{G}_1$ to a_b^* and sends it to $\mathcal{B}$.

The reduction $\mathcal{B}$ uses the received commitment to create the shares of credential for $\mathcal{A}$, and its own partial shares of credential because it does not know the shares of the attribute a_b^* committed to by $\mathcal{C}$.

During the presentation protocol queries the reduction $\mathcal{B}$ simulates the execution of the presentation protocol programming the random oracle.

At the end of the training, the adversary $\mathcal{A}$ outputs a bit b' specifying their guess about the attribute included in the credential, and $\mathcal{B}$ forwards b' to $\mathcal{C}$.

7.4 Presentation with identifiable abort of BBS MHAC

Lemma 5. *Assuming that the protocol participants communicate over an authenticated channel, $\mathcal{H}_{\mathsf{com}}$ is a secure commitment scheme, $\Pi_{\mathsf{MHAC-BBS}}$ satisfies presentation with identifiable abort (Definition 11).*

We sketch the proof of Lemma 5. A more detailed proof is given in Appendix E.

Proof Sketch. We observe that Protocol 3 already describes the composition of a presentation protocol with the associated identifiable abort detector algorithm W that manifests itself in Step 5 and Step 7. It is easy to see that correctness holds because if W, which is already included in Protocol 3, does not output $\perp$ at the end of Step 5, the participants have created a shared first message U and can compute a shared challenge ch. Then if every participant P_i correctly proves knowledge of a representation (w.r.t. the prescribed bases) of $\tilde{B}_i$, these proofs can be aggregated, leading to a proof of knowledge of a representation of $\overline{B}$. Concerning identifiability, it is easy to see that if the algorithm W outputs (abort, i), the party P_i has misbehaved, either because it has not opened the commitment to U_i correctly, or because it has not created a valid zero-knowledge proof of $\tilde{B}_i$.

7.5 Unforgeability of presentations of BBS MHAC.

Lemma 6. *Assuming* BBS *is SUF-CMA and the* DL *assumption holds in our group, $\Pi_{\mathsf{MHAC-BBS}}$ satisfies concurrent unforgeability of presentations (Definition 12) against an active static adversary corrupting less than t holders and an honest-but-curious issuer.*

We sketch the security proof of Lemma 6 and we provide a complete proof in Appendix F.

Proof Sketch. To prove that $\Pi_{\mathsf{MHAC-BBS}}$ is unforgeable according to the security notion of Definition 12, we instantiate the unforgeability experiment $\mathsf{Exp}^{\mathsf{c-uf-pres}}_{\mathcal{F}}(\kappa)$ in the case of BBS MHAC in Appendix F.1, which results in the definition of $\mathsf{Exp}^{\mathsf{c-uf-pres}}_{\mathcal{F},\mathsf{BBS}}(\kappa)$. Then, we show how it is possible to use an adversary $\mathcal{F}$ of the experiment $\mathsf{Exp}^{\mathsf{c-uf-pres}}_{\mathcal{F},\mathsf{BBS}}(\kappa)$ as a subroutine of a reduction $\mathcal{B}$ to the DL assumption, if the adversary forges a presentation derived from the target credential (Case A), or to the qSDH assumption, if the adversary forges a presentation derived from another credential it was never issued (Case B). More precisely, a reduction that rewinds the adversary $\mathcal{F}$ will end up extracting, from the adversary's forgeries (that are proofs of knowledge of a BBS credential) a credential that will fall into one of these two cases (as we show in Appendix F.2). It is easy to see that, for MHAC schemes compatible with secure anonymous credential schemes this proving Case B is trivial, since it is possible to easily reduce to the unforgeability of the digital signature scheme underlying the anonymous credential scheme.

Proving Case A instead is more challenging, and in this sketch proof we limit to describe how our reduction can set up the unforgeability experiment to reduce the DL assumption.

We consider a forger $\mathcal{F}$ who can forge a presentation associated to the target credential it is issued.

We must define a reduction $\mathcal{B}$ interacting with $\mathcal{F}$, and with the challenger $\mathcal{C}_{\mathsf{DL}}$ of the DL problem (Definition 1), who can win the DL experiment with non-negligible probability, if $\mathcal{F}$ wins the unforgeability experiment with non-negligible probability.

The reduction $\mathcal{B}$ receives in input the tuple $(p, \mathbb{G}_1, g, h)$ from $\mathcal{C}_{\mathsf{DL}}$, where $(g, h) \in \mathbb{G}_1^2$ is an instance of the discrete logarithm problem that $\mathcal{B}$ needs to solve.

Setup Phase. $\mathcal{B}$ must generate the public parameters to send to $\mathcal{F}$, and the issuer's public key for the BBS signature scheme. It must generate it in a way that, when $\mathcal{F}$ sends an issuance query $(\{a_i\}_{i\in[m]}, t, n, \mathsf{cor})$ for the target credential, it will be able to generate $t-1$ shares of the target credential for the parties in cor corrupted by $\mathcal{F}$

$$\{\mathsf{cred}_i\}_{i\in\mathsf{cor}} \leftarrow ((A, \{e^{\star(i)}\}_{i\in\mathsf{cor}}, \{D_i\}_{i\in[n]}, \{a_j\}_{j\in[m]}),$$

which is a secret sharing of a BBS credential $((A, e^\star), \{a_i\}_{i\in[m]})$ where the value $e^\star = \log_g h$, and is unknown to $\mathcal{B}$[17]. In particular, $\mathcal{B}(g, h)$ must generate pp, x in a way that, for any $\{a_i\}_{i\in[m]} \in \mathbb{Z}_p^m$, it will be able to compute the value $A = C(\mathbf{a})^{\frac{1}{x+e^\star}}$, which is univocally determined by the attributes once DL challenge (g, h) and pp, x are fixed. Additionally, $\mathcal{B}$ must be able to generate $D = A^{-e^\star}$

[17] We recall that in this experiment we do not consider private attributes because the challenger always learns the attributes from the online-extractable proofs π it receive from the holders in the issuing protocol (Protocol 2).

that is secret shared in $\{D_i\}_{i\in[n]}$ which is implicitly included in every share of credential.

To do that, $\mathcal{B}$ performs the following operations:

1. samples the group generator of $\mathbb{G}_2$, $g_2 \xleftarrow{\$} \mathbb{G}_2$, the issuer's secret key $x \xleftarrow{\$} \mathbb{Z}_p$, and sets $X_2 = g_2^x$ as in Algorithm 1;
2. sets $k \leftarrow g^x h, \quad k \in \mathbb{G}_1$, which is the trapdoor that allows $\mathcal{B}$ to compute, $\forall \mathbf{a} = \{a_i\}_{i\in[m]} \in \mathbb{Z}_p^m$ the values $A, A^{-e^\star}$ satisfying $A^{x+e^\star} = C(\mathbf{a})$;
3. generates the public parameters pp as follows: $\gamma_0, \gamma_1, \ldots, \gamma_m \xleftarrow{\$} \mathbb{Z}_p$ then, set $g_1 \leftarrow k^{\gamma_0}$ as the generator of $\mathbb{G}_1$ and $h_i = k^{\gamma_i}, \forall i \in [m]$ and $\mathsf{pp} \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2, h_1, \ldots, h_m)$;
4. sends pp, X_2 to $\mathcal{F}$.

The simulation of parameter generation and key generation is indistinguishable from a real execution of the parameter generation because the key generation is calculated exactly the same way, and the elements $(g_2, g_1, h_1, \ldots, h_m)$ are chosen uniformly at random in $\mathbb{G}_2 \times \mathbb{G}_1^{m+1}$. However, $\mathcal{B}$ knows the discrete logarithm of the elements in G_1 with respect to the basis $k = g^x h$.

Training Phase. During the Training Phase the adversary $\mathcal{F}$ we consider in Case A will send an issuance query for the single target credential, giving in input to $\mathcal{O}_{\mathsf{pres}}$ the tuple $(\{a_i\}_{i\in[m]}, t, n, \mathsf{cor}), |\mathsf{cor}| = t - 1$.
Without loss of generality we can restrict to the case $t = n$, and $\mathsf{cor} = [t-1]$.

Having received $(\{a_i\}_{i\in[m]}, t, n, \mathsf{cor})$ from $\mathcal{F}$, $\mathcal{B}$ computes $\alpha = \log_k C(\mathbf{a})$, which is

$$\alpha = \gamma_0 \sum_{i=1}^{m} \gamma_i a_i.$$

Being $k = g^x h = g^{x+e^\star}$, for the unknown $e^\star$, the knowledge of α allows $\mathcal{B}$ to compute

$$A = C(\mathbf{a})^{\frac{1}{x+e^\star}} = (k^\alpha)^{\frac{1}{x+e^\star}} = (k^{\frac{1}{x+e^\star}})^\alpha = g^\alpha,$$
$$A^{e^\star} = (g^\alpha)^{e^\star} = (g^{e^\star})^\alpha = h^\alpha.$$

In summary, $\mathcal{B}$ simulates the issuance of the target credential as follows:

1. computes $\alpha \leftarrow \gamma_0 \sum_{i=1}^{m} \gamma_i a_i$ and sets $A \leftarrow g^\alpha$ and $D \leftarrow (h^\alpha)^{-1}$;
2. simulates a secret sharing of $e^\star$: $\{e^{\star(i)}\}_{i\in\mathsf{cor}} \xleftarrow{\$} \mathbb{Z}_p^{|\mathsf{cor}|}$;
3. sets $D_i \leftarrow A^{-e^{\star(i)}}, \forall i \in \mathsf{cor}$ and $D_n \leftarrow D \prod_{i\in\mathsf{cor}} D_i^{-1}$;
4. $\mathcal{B}$ sets $\{\mathsf{cred}_i\}_{i\in\mathsf{cor}} \leftarrow ((A, \{e^{\star(i)}\}_{i\in\mathsf{cor}}, \{D_i\}_{i\in[n]}, (\{a_j\}_{j\in[m]}))$.

This completes the simulation of the issuance of the target credential.

Note that $\mathcal{B}$ knows all the information associated with the target credential apart from the value $e^{\star(n)} = -\log_A D_n$.

When $\mathcal{F}$ sends to $\mathcal{B}$ a query to create a presentation of the target credential, with input $(\mathsf{nonce}, \{a_i\}_{i\in\mathsf{Rev}}, \mathsf{hon})$, we can assume that $\mathcal{F}$ controls the primary

party who sends to $\mathcal{B}$ the value $r \in \mathbb{Z}_p$. Then $\mathcal{B}$ can compute $\overline{A} = A^r, \overline{B} = C(\mathbf{a})^r \overline{A}^{-e^\star} = C(\mathbf{a})^r D^r$ and $\tilde{B}_n = D_n^r$. Given this information, $\mathcal{B}$ can simulate the presentation protocol by programming the random oracle similarly to how it is done in [CKM23a].

We include all the remaining details of the simulation of Case A, and the whole analysis of Case B, in Appendix F.2.

We highlight that our reduction can simulate the unforgeability experiment without rewinding the adversary, therefore the reductions both to the DL assumption and to the qSDH assumption[18] described in Appendix F.2 allow the adversary to open concurrent presentation session during the training phase. This guarantees the concurrent security of the BBS multi-holder anonymous credential scheme $\Pi_{\mathsf{MHAC-BBS}}$.

Acknowledgments. Andrea Flamini acknowledges support from Eustema S.p.A. through the PhD scholarship and is supported by the QUBIP project, funded by the European Union under the Horizon Europe framework program [grant agreement no. 101119746]. Eysa Lee was supported by the Data Science Institute at Brown University. Anna Lysyanskaya was supported by NSF Grants 2312241, 2154170, and 2247305, as well as funding from a Brown University Seed Award and Meta.

References

AAM23. Mobile Driver's License (mDL) implementation guidelines, version 1.2. https://www.aamva.org/topics/mobile-driver-license, 01 2023.

ASM06. Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic k-TAA. In *SCN 2006*, volume 4116 of *LNCS*, pages 111–125, 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, 2008.

BBC+24. Carsten Baum, Olivier Blazy, Jan Camenisch, Jaap-Henk Hoepman, Eysa Lee, Anja Lehmann, Anna Lysyanskaya, René Mayrhofer, Hart Montgomery, Ngoc Khanh Nguyen, Bart Praneel, abhi shelat, Daniel Slamanig, Stefano Tessaro, Søren Eller Thomsen, and Carmela Troncoso. Cryptographers' feedback on the eu digital identity's ARF. https://github.com/user-attachments/files/15904122/cryptographers-feedback.pdf, 2024.

BBS04. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Annual international cryptology conference*, pages 41–55. Springer, 2004.

BF24. Michele Battagliola and Andrea Flamini. Distributed fiat-shamir transform: from threshold identification protocols to signatures. *Cryptology ePrint Archive*, 2024.

BL10. Ernie Brickell and Jiangtao Li. A pairing-based daa scheme further reducing tpm resources. In *International Conference on Trust and Trustworthy Computing*, pages 181–195. Springer, 2010.

[18] The strong unforgeability of BBS signatures is proven to hold in [TZ23] under the qSDH assumption.

BLT+24. Renas Bacho, Julian Loss, Stefano Tessaro, Benedikt Wagner, and Chenzhi Zhu. Twinkle: Threshold signatures from ddh with full adaptive security. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 429–459. Springer, 2024.

BN06. Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 390–399, 2006.

BS23. Dan Boneh and Victor Shoup. A graduate course in applied cryptography. *Draft 0.6*, 2023.

CDG+24. Megan Chen, Pousali Dey, Chaya Ganesh, Pratyay Mukherjee, Pratik Sarkar, and Swagata Sasmal. Universally composable non-interactive zero-knowledge from sigma protocols via a new straight-line compiler. *Cryptology ePrint Archive*, 2024.

CDL16. Jan Camenisch, Manu Drijvers, and Anja Lehmann. Anonymous attestation using the strong diffie hellman assumption revisited. In *Trust and Trustworthy Computing: 9th International Conference, TRUST 2016, Vienna, Austria, August 29-30, 2016, Proceedings 9*, pages 1–20. Springer, 2016.

CKLS02. Christian Cachin, Klaus Kursawe, Anna Lysyanskaya, and Reto Strobl. Asynchronous verifiable secret sharing and proactive cryptosystems. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 88–97, 2002.

CKM23a. Elizabeth Crites, Chelsea Komlo, and Mary Maller. Fully adaptive schnorr threshold signatures. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 678–709, Cham, 2023. Springer Nature Switzerland.

CKM23b. Elizabeth Crites, Chelsea Komlo, and Mary Maller. Fully adaptive schnorr threshold signatures. *Cryptology ePrint Archive*, 2023.

CL01. Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer, 2001.

CL03. Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In *Security in Communication Networks: Third International Conference, SCN 2002 Amalfi, Italy, September 11–13, 2002 Revised Papers 3*, pages 268–289. Springer, 2003.

CL04. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Annual international cryptology conference*, pages 56–72. Springer, 2004.

CL17. Ran Cohen and Yehuda Lindell. Fairness versus guaranteed output delivery in secure multiparty computation. *Journal of Cryptology*, 30(4):1157–1186, 2017.

CLOS02. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 494–503, 2002.

Dam02. Ivan Damgård. On σ-protocols. *Lecture Notes, University of Aarhus, Department for Computer Science*, 84, 2002.

DKL+23. Jack Doerner, Yashvanth Kondi, Eysa Lee, abhi shelat, and LaKyah Tyner. Threshold bbs+ signatures for distributed anonymous credential issuance. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 773–789. IEEE, 2023.

Fis05. Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *Annual International Cryptology Conference*, pages 152–168. Springer, 2005.

FSS+24. Andrea Flamini, Giada Sciarretta, Mario Scuro, Amir Sharif, Alessandro Tomasi, and Silvio Ranise. On cryptographic mechanisms for the selective disclosure of verifiable credentials. *Journal of Information Security and Applications*, 83:103789, 2024.

GPS08. Steven D Galbraith, Kenneth G Paterson, and Nigel P Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

HS21. Lucjan Hanzlik and Daniel Slamanig. With a little help from my friends: Constructing practical anonymous credentials. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2004–2023, 2021.

HSS23. Julia Hesse, Nitin Singh, and Alessandro Sorniotti. How to bind anonymous credentials to humans. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 3047–3064, Anaheim, CA, August 2023. USENIX Association.

IOZ14. Yuval Ishai, Rafail Ostrovsky, and Vassilis Zikas. Secure multi-party computation with identifiable abort. In *Advances in Cryptology–CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II 34*, pages 369–386. Springer, 2014.

KL07. Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography: principles and protocols*. Chapman and hall/CRC, 2007.

KMR12. Marcel Keller, Gert Læssøe Mikkelsen, and Andy Rupp. Efficient threshold zero-knowledge with applications to user-centric protocols. In *Information Theoretic Security: 6th International Conference, ICITS 2012, Montreal, QC, Canada, August 15-17, 2012. Proceedings 6*, pages 147–166. Springer, 2012.

KS22. Yashvanth Kondi and Abhi Shelat. Improved straight-line extraction in the random oracle model with applications to signature aggregation. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 279–309. Springer, 2022.

LHAT20. Wouter Lueks, Brinda Hampiholi, Greg Alpár, and Carmela Troncoso. Tandem: Securing keys by using a central server while preserving privacy. *Proceedings on Privacy Enhancing Technologies*, 2020:327–355, 07 2020.

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.

LR22. Anna Lysyanskaya and Leah Namisa Rosenbloom. Universally composable σ-protocols in the global random-oracle model. In *Theory of Cryptography Conference*, pages 203–233. Springer, 2022.

MY24. Jamal H Mosakheil and Kan Yang. Silentproof: Anonymous authentication with blockchain-backed offloading. In *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*, pages 1361–1377, 2024.

Pas03. Rafael Pass. On deniability in the common reference string and random oracle model. In *Advances in Cryptology-CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings 23*, pages 316–337. Springer, 2003.

Ped91. Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual international cryptology conference*, pages 129–140. Springer, 1991.

PS96. David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *International conference on the theory and applications of cryptographic techniques*, pages 387–398. Springer, 1996.

PS16. David Pointcheval and Olivier Sanders. Short randomizable signatures. In *Topics in Cryptology-CT-RSA 2016: The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29-March 4, 2016, Proceedings*, pages 111–126. Springer, 2016.

Sha79. Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

TZ23. Stefano Tessaro and Chenzhi Zhu. Revisiting bbs signatures. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 691–721. Springer, 2023.

Unr15. Dominique Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In *Advances in Cryptology-EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II 34*, pages 755–784. Springer, 2015.

A Sigma Protocol for Linear Relation

We reproduce a sigma protocol for linear relations [BS23, Figure 19.8] in Figure 1, which will be used as a building block for our construction. An extensive description of this protocol and related ones can be found in [BS23, Section 19.5].

Letting $\mathsf{pp} \leftarrow (\mathbb{G}, p, g_1, \ldots, g_m)$, Figure 1 can be used to prove knowledge of a representation of a statement $h \in \mathbb{G}$ with respect to the bases $g_1, \ldots, g_n \in \mathbb{G}$ for the relation

$$\mathcal{R}_{\mathsf{LR}}(\mathsf{pp}) = \{(x_1, \ldots, x_m, h) \in \mathbb{Z}_p^n \times \mathbb{G} : \prod_{i=1}^{m} g_i^{x_i} = h\}.$$

Theorem 4 ([BS23, Theorem 19.11]). *The generic linear protocol in Figure 1 is a Sigma protocol for the relation $\mathcal{R}_{\mathsf{LR}}$. Moreover, it provides special soundness and is special HVZK.*

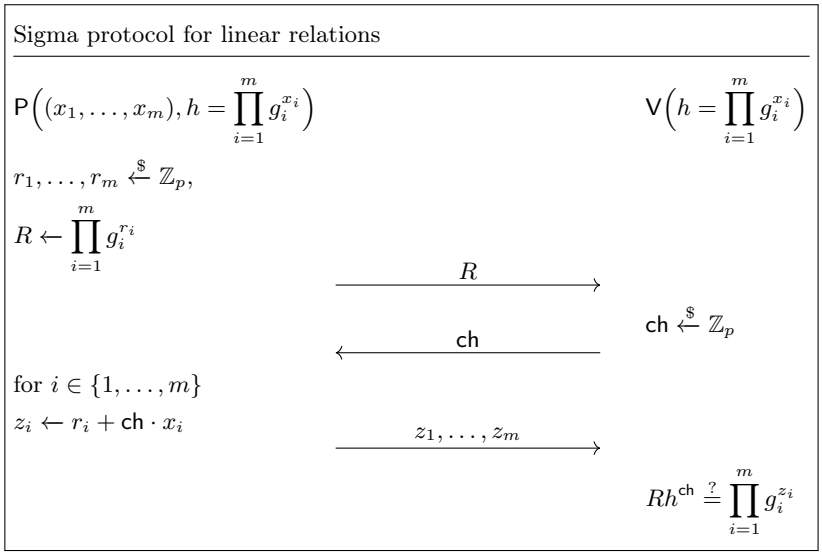

Fig. 1. Sigma protocol for $\mathcal{R}_{\mathsf{LR}}(\mathsf{pp})$. The public parameters pp defining the relation are $p, \mathbb{G}, g_1, \ldots, g_m$.

B Analysis of BBS Presentation Protocol in [TZ23].

In this section we focus on the security analysis of Protocol 4 proposed in [TZ23]. The protocol is obtained by applying the Fiat-Shamir transform to the sigma protocol in Figure 2. It easy to see that this sigma protocol satisfies the completeness property, we recall how it is possible to prove the HVZK property and we extensively discuss the special soundness property that the protocol satisfies, because it requires some caveat based on the use of the sigma protocol.

Protocol 4. Presentation of a BBS credential of [TZ23]

Signature material randomization phase. The holder
- sets $C(\mathbf{a}) \leftarrow g_1 \prod_{i=1}^{m} h_i^{a_i}$ and $C(\mathbf{a}') \leftarrow g_1 \prod_{i \in \mathsf{Rev}} h_i^{a_i}$
- samples uniformly at random $r \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\overline{A} \leftarrow A^r$$
$$\overline{B} \leftarrow C(\mathbf{a})^r \overline{A}^{-e} = C(\mathbf{a}')^r \Big(\prod_{i \in \mathsf{Hid}} h_i^{r a_i} \Big) \overline{A}^{-e}$$

Sigma protocol execution phase. The holder must prove knowledge of a representation of $\overline{B}$ w.r.t. the bases $C(\mathbf{a}'), \{h_i\}_{i \in \mathsf{Hid}}, \overline{A}$.

To do so, given the statement $\overline{B}$ the holder executes the sigma protocol for the linear relation defined by the following parameters:

$$\mathcal{R}_{\mathsf{LR}}(\mathbb{G}_1, p, C(\mathbf{a}'), \{h_i\}_{i\in\mathsf{Hid}}, \overline{A}) = \\ = \Big\{ \Big(\underbrace{(x, \{y_j : j \in \mathsf{Hid}\}, z)}_{\text{witness}}, \underbrace{\overline{B}}_{\text{statement}} \Big) : \overline{B} = C(\mathbf{a}')^x \Big(\prod_{j\in\mathsf{Hid}} h_j^{y_j} \Big) \overline{A}^z \Big\} \quad (2)$$

with $\mathcal{R}_{\mathsf{LR}}(\mathbb{G}_1, p, C(\mathbf{a}'), \{h_i\}_{i\in\mathsf{Hid}}, \overline{A}) \subset \mathbb{Z}_p^{|\mathsf{Hid}|+2} \times \mathbb{G}_1$. The prover will set the witness as the tuple $(r, \{ra_j : j \in \mathsf{Hid}\}, -e)$ for the statement $\overline{B}$ [19].

Interactive BBS credential presentation of [TZ23]

Prover $\mathsf{P}\Big((A, e, \{a_i\}_{i\in\mathsf{Hid}}), (\{a_j\}_{j\in\mathsf{Rev}})\Big)$		**Verifier** $\mathsf{V}\Big(\{a_j\}_{j\in\mathsf{Rev}}\Big)$
$r \xleftarrow{\$} \mathbb{Z}_p, \ \ \overline{A} \leftarrow A^r, \ \ \overline{B} \leftarrow C(\mathbf{a})^r \overline{A}^{-e}$		
$v_r, \{v_j\}_{j\in\mathsf{Hid}}, v_e \xleftarrow{\$} \mathbb{Z}_p,$		
$U \leftarrow C(\mathbf{a}')^{v_r} \Big(\prod_{j\in\mathsf{Hid}} h_j^{v_j} \Big) \overline{A}^{v_e}$	$\xrightarrow{\overline{A}, \overline{B}, U}$	
	$\xleftarrow{\mathsf{ch}}$	$\mathsf{ch} \xleftarrow{\$} \mathbb{Z}_p$
$z_r \leftarrow v_r + \mathsf{ch} \cdot r$		
$z_j \leftarrow v_j + \mathsf{ch} \cdot ra_j, \ \ \forall j \in \mathsf{Hid}$		
$z_e \leftarrow v_e + \mathsf{ch} \cdot (-e)$		
$\mathsf{rsp} \leftarrow (z_r, \{z_j\}_{j\in\mathsf{Hid}}, z_e)$	$\xrightarrow{\mathsf{rsp}}$	
		$U\overline{B}^{\mathsf{ch}} \stackrel{?}{=} C(\mathbf{a}')^{z_r} \left(\prod_{j\in\mathsf{Hid}} h_j^{z_j} \right) \overline{A}^{z_e}$
		$\mathbf{e}(\overline{A}, X_2) \stackrel{?}{=} \mathbf{e}(\overline{B}, g_2)$

Fig. 2. Sigma protocol for $R_{\mathsf{BBS}}(\mathsf{pp}, X_2)$. We recall that $C(\mathbf{a}') \leftarrow g_1 \prod_{i\in\mathsf{Rev}} h_i^{a_i}$ and $C(\mathbf{a}) \leftarrow \prod_{i=1}^m h_i^{a_i}$.

Honest Verifier Zero-Knowledge. Assuming that the signer publishes a pair $(U, U^x) \in \mathbb{G}_1^2$, then it is also possible to simulate the interaction of a prover

with an honest verifier[20]. In fact, a simulator can simulate the signature material randomization phase by sampling $r \xleftarrow{\$} \mathbb{Z}_p$ and computing $\overline{A} \leftarrow U^r$ and $\overline{B} \leftarrow (U^x)^r$ then by simulating the sigma protocol execution phase, which can be simulated since there exist a simulator for the sigma protocol for linear relations (see Appendix A), we obtain a simulator for the whole interactive protocol.

Special soundness. Regarding the special soundness property, observe that the sigma protocol for linear relation is a knowledge sound sigma protocol, therefore from two transcripts for the same first message and different challenge it is possible to extract a witness for the statement $\overline{B}$ under the relation $\mathcal{R}(\overline{A}, \mathsf{Hid}, \mathsf{pp})$, namely the witness $(\rho, \{\mu_j : j \in \mathsf{Hid}\}, \varepsilon)$[21].

Now we must distinguish two cases:

- $\rho \neq 0$: in this case the extractor can extract a valid BBS signature $(A, e = -\epsilon)$ for the messages in $\mathbf{a}'' = \{a_i : i \in \mathsf{Rev}\} \cup \{\frac{\mu_j}{\rho} : j \in \mathsf{Hid}\}$. In fact it holds that

$$\overline{B} = C(\mathbf{a}')^{\rho} \left(\prod_{i \in \mathsf{Hid}} h_i^{\mu_i} \right) \overline{A}^{\varepsilon} \implies \overline{B}^{\frac{1}{\rho}} = C(\mathbf{a}') \left(\prod_{i \in \mathsf{Hid}} h_i^{\frac{\mu_i}{\rho}} \right) \overline{A}^{\frac{\varepsilon}{\rho}} = C(\mathbf{a}'') \overline{A}^{\frac{\varepsilon}{\rho}} \tag{3}$$

  and also that

$$\mathbf{e}(\overline{A}, X_2) = \mathbf{e}(\overline{B}, g_2) \implies \mathbf{e}(\overline{A}^{\frac{1}{\rho}}, X_2) = \mathbf{e}(\overline{B}^{\frac{1}{\rho}}, g_2) = \mathbf{e}(C(\mathbf{a}'')(\overline{A}^{\frac{1}{\rho}})^{\varepsilon}, g_2) \tag{4}$$

  where the last equality holds by Equation 3. Combining Equation 4 and Equation 1, by setting $A \leftarrow \overline{A}^{\frac{1}{\rho}}$ and $e \leftarrow -\varepsilon$ the extractor extracts a signature (A, e) for the messages in $\mathbf{a}''$. This contradicts the unforgeability of the BBS signature and therefore the qSDH assumption.
- $\rho = 0$: in such case the extractor can not extract a valid signature for the messages in $\mathbf{a}''$, but
  - if $\mu_i = 0, \forall i \in \mathsf{Hid}$ then the extractor extracts the secret key of the signer $x = \epsilon$, since $\overline{B} = \overline{A}^{\epsilon}$ and $\mathbf{e}(\overline{B}, g_2) = \mathbf{e}(\overline{A}, X_2)$. Given the secret key of the issuer the extractor can "forge" as many signatures as it wants;
  - if $\mu_i \neq 0$ for some $i \in \mathsf{Hid}$ then the extractor can compute $\overline{B} = \left(\prod_{i \in \mathsf{Hid}} h_i^{\mu_i}\right) \overline{A}^{\epsilon}$. But

$$\mathbf{e}(\overline{A}, X_2) = \mathbf{e}(\overline{B}, g_2) \implies \mathbf{e}(\overline{A}, X_2) = \mathbf{e}(\left(\prod_{i \in \mathsf{Hid}} h_i^{\mu_i} \right) \overline{A}^{\epsilon}, g_2),$$

[20] Note that pairs of random elements (S, T) in $\mathbb{G}_1$ can not be assumed indistinguishable from pairs of elements of the form (U, U^x) since it is always possible to check if $\mathbf{e}(S, X_2) = \mathbf{e}(T, g_2)$

[21] Note that this setting is different from the one described in Section A, in fact the prover must prove knowledge of the representation of a statement with respect to a set of bases which depend on the statement ($\overline{A}$ depends on $\overline{B}$).

which means that

$$\mathbf{e}(\overline{A}, X_2 g_2^e) = \mathbf{e}(\prod_{i \in \mathsf{Hid}} h_i^{\mu_i}, g_2)$$

therefore the extractor can extract a tuple $(\overline{A}, e, \{\mu_i\}_{i \in \mathsf{Hid}})$ such that $\overline{A} = \left(\prod_{i \in \mathsf{Hid}} h_i^{\mu_i}\right)^{\frac{1}{x+e}}$. We refer to such a tuple as *wildcard credential*[22].

This is not a BBS signature on a set of attributes therefore this protocol is not a proper special sound sigma protocol to prove knowledge of a BBS credential but a proof of knowledge of a BBS credential or a wildcard credential. More formally, it is a sigma protocol for the following relation.

$$\begin{aligned}
&\mathcal{R}(\mathsf{pp}, X_2) = \\
&= \Big\{ \Big(\underbrace{(A, e, \{a_i\}_{i \in \mathsf{Hid}})}_{\text{witness}}, \underbrace{\{a_j\}_{j \in \mathsf{Rev}}}_{\text{statement}} \Big) : \mathbf{e}(A, X_2 g_2^e) = \mathbf{e}\Big(g_1 \prod_{i \in \mathsf{Hid}} h_i^{a_i} \prod_{j \in \mathsf{Rev}} h_j^{a_j}, g_2\Big) \\
&\quad \vee \mathbf{e}(A, X_2 g_2^e) = \mathbf{e}(\prod_{i \in \mathsf{Hid}} h_i^{a_i}, g_2) \Big\}
\end{aligned} \tag{5}$$

Now we argue why this sigma protocol can be used to create proofs of knowledge of BBS credentials.

Observation 6. *In an anonymous credential system we must assume that the issuer is, in the worst case, a passive adversary (i.e. honest but curious). This means that the issuer only issues well formed credentials of the form* $((A, e), \{a_i\}_{i \in [m]})$ *satisfying* $\mathbf{e}(A, X_2 g_2^e) = \mathbf{e}\Big(g_1 \prod_{i \in \mathsf{Hid}} h_i^{a_i} \prod_{j \in \mathsf{Rev}} h_j^{a_j}, g_2\Big)$. *Therefore, if it were possible to show that an adversary who can see only* BBS *signatures (the only information that the issuer gives to the other actors in the ecosystem) can not forge a wildcard credential, then it would be possible to be assured that valid presentations are proofs of possession of a* BBS *credential.*

In [TZ23] the authors prove the strong unforgeability under chosen message attacks of the BBS signature under the qSDH assumption. This means that an adversary who can see a polynomial number of BBS signatures for messages of its choice (i.e. BBS credentials) can not create a signature different from the ones it has seen before under the qSDH assumption. Using a similar argument, it is possible to prove that an adversary who can see a polynomial number of BBS signatures for messages of its choice not only it can not forge a BBS signature, but *it can not forge even a wildcard credential* $((A, e), \{a_i\}_{i \in \mathsf{Hid}})$ under the qSDH assumption. The proof of this statement can be obtained by adapting the proof of unforgeability of BBS signatures in [TZ23, Section 3] by considering an adversary producing as forgery a wildcard credential, and show that a reduction can use such forgery to break the qSDH assumption. The simulation of the unforgeability experiment is identical, the only change required to the security proof is

[22] We call it wildcard credential because this credential would allow the holder to present any attribute in the positions $j \notin \mathsf{Hid}$ setting the exponent of $C(\mathbf{a}')$ to 0

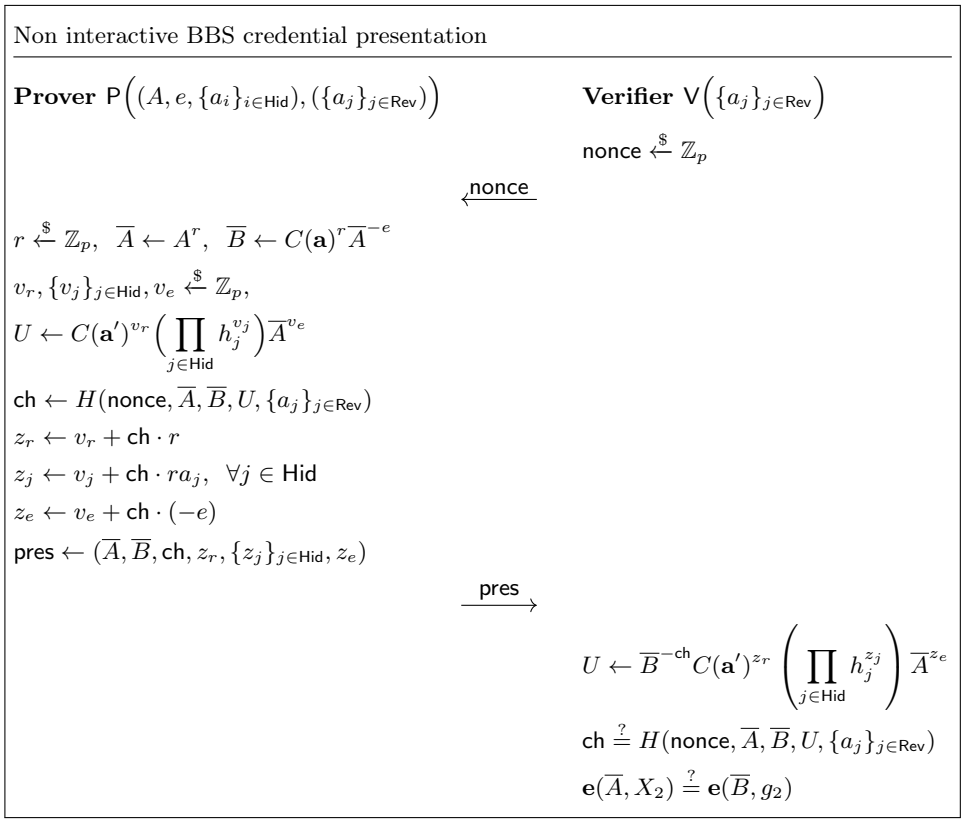

Fig. 3. Signature of the nonce provided by the verifier to the holder to generate a proof of knowledge of a BBS credential issued by $\mathsf{pk} = X_2$.

the way the reduction retrieves the solution to the qSDH experiment from the forgery of the adversary, which in this case is a wildcard credential.

This is the reason why the sigma protocol described in Protocol 4 can be used to prove knowledge of a BBS credential. Since we must assume that the issuer is not actively corrupt can be used to generate proof of knowledge of a BBS credentials under the qSDH assumption.

Protocol for fresh presentations. Below, in Figure 3, we describe the algorithms for the creation and verification of a fresh BBS presentation computed using the sigma protocol made non-interactive applying the Fiat-Shamir transform generating the challenge using the nonce that the holder has received from the verifier.

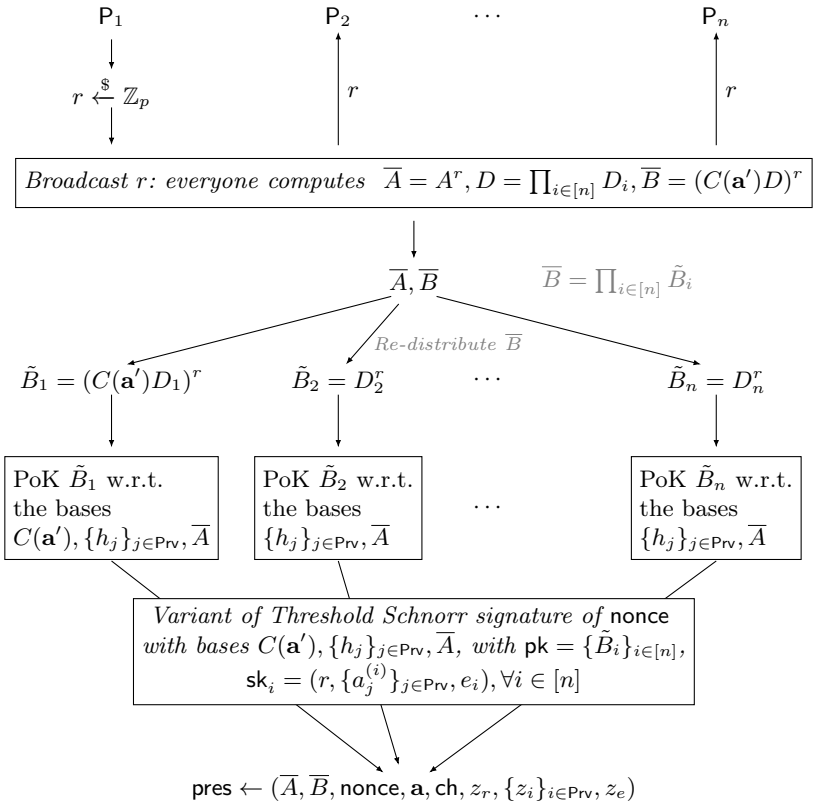

Fig. 4. Presentation protocol overview for the simplified case of (1)full disclosure of the attributes $\{a_j\}_{j\in\mathsf{Pub}}$ and (2) full threshold, i.e. $t = n$. In this example the primary party is P_1.

C BBS MHAC Presentation Protocol Overview

In Figure 4 we sketch the presentation protocol executed by a set of holders $\{\mathsf{P}_i\}_{i\in[n]}$ of a $(n, n) - \mathsf{BBS}$ multi-holder anonymous credential. The inputs of each party P_i are:

- $\mathsf{cred}_i = (A, e^{(i)}, \{D_k\}_{k\in[n]\setminus\{i\}}, \{a_k^{(i)}\}_{k\in\mathsf{Prv}}, \{a_k\}_{k\in\mathsf{Pub}})$
- $\mathbf{a}' = \{a_k\}_{k\in\mathsf{Rev}}$
- the presentation nonce;

We recall that $D_k = \prod_{j\in\mathsf{Prv}} h_j^{a_j^{(k)}} A^{-e^{(k)}}$.

The boxes representing the PoK of $\tilde{B}_i$ describe the operations that each holder executes in the variant of Threshold Schnorr signature, it is not a separate step. Also, in the security proof, the reduction does not have to extract the secret shares of all the holders controlled by the adversary (otherwise it would be a), but only the BBS credential used by the adversary to generate its forgery.

D Private Attribute Unlinkability of BBS MHAC Scheme

In this section we formally prove Lemma 4 that states that the BBS MHAC scheme satisfies the unlinkability of private attributes defined in Definition 9. For the sake of simplicity we consider the full threshold case, i.e. $t = n$ and an adversary who corrupts the parties in $[t-1]$.

Proof. We define a reduction $\mathcal{B}$ to the hiding property of the Pedersen commitments which are perfectly hiding. The reduction works as follows:

1. The reduction $\mathcal{B}$ sends to the adversary $\mathcal{A}$ a set of public parameters for the BBS MHAC scheme and an issuer key pair $(x, g_2^x), x \xleftarrow{\$} \mathbb{Z}_p$.
2. The adversary $\mathcal{A}$ of the private attribute unlinkability sends to $\mathcal{B}$ two attributes a_0^*, a_1^* and the set of public attributes $\{a_i\}_{i\in\mathsf{Pub}}$.
3. $\mathcal{B}$ sends the same messages a_0^*, a_1^* to the challenger $\mathcal{C}$ of the hiding property of the Pedersen commitment who samples a bit b uniformly at random and computes a Pedersen commitment of a_b^*, namely $D = h_{m-1}^{a_b^*} h_m^s \in \mathbb{G}_1$ for $s \xleftarrow{\$} \mathbb{Z}_p$ and sends it to $\mathcal{B}$.
4. $\mathcal{B}$ generates a BBS credential $\mathsf{cred} = (A, e)$ with $e \xleftarrow{\$} \mathbb{Z}_p$ and
$$A = \left(g_1 \big(\prod_{i\in\mathsf{Pub}} h_i^{a_i} \big) D \right)^{\frac{1}{x+e}}.$$
And from cred generates the shares of credential for $\mathcal{A}$ (controlling the parties $\{\mathsf{P}_i\}_{i\in[n-1]}$):
   - $\mathcal{B}$ samples, $(a_b^{*(i)}, s^{(i)}, e^{(i)}) \xleftarrow{\$} \mathbb{Z}_p^3, \forall i \in [n-1]$;
   - sets $D_i = h_{m-1}^{a_b^{*(i)}} h_m^{s^{(i)}} A^{-e^{(i)}}$, $D_n \leftarrow \frac{D}{\prod_{i\in[n-1]} D_i}$, and $e^{(n)} \leftarrow e - \sum_{i\in[n-1]} e^{(i)}$.
   - sets for all $i \in [n-1]$
$$\mathsf{cred}_i = \left(A, e^{(i)}, \{D_k\}_{k\in[n]\setminus\{i\}}, (\{a_j\}_{j\in\mathsf{Pub}}, a_b^{*(i)}, s^{(i)}) \right),$$
   and
$$\mathsf{cred}_n = \left(A, e^{(n)}, \{D_k\}_{k\in[n]\setminus\{n\}}, (\{a_j\}_{j\in\mathsf{Pub}}, \perp, \perp) \right).$$
   - $\mathcal{B}$ sends the shares cred_i to $\mathcal{A}$.
5. $\mathcal{A}$ sends queries to perform presentation protocol executions giving in input the values nonce and Rev of its choosing. We informally describe the simulation of the presentation queries. $\mathcal{B}$ executes the presentation protocol, but when it comes the time to execute Step 3 of the presentation protocol, since it does not know a representation of D_n w.r.t. h_{m-1}, h_m, A. $\mathcal{B}$ performs the following operations:
   - it simulates a sigma protocol transcript for the proof of knowledge of a representation of D_n, picking $\mathsf{ch}, z_{m-1}^{(n)}, z_m^{(n)}, z_e^{(n)} \xleftarrow{\$} \mathbb{Z}_p$ and setting $U_n \leftarrow \tilde{B}_n^{-\mathsf{ch}} h_{m-1}^{z_{m-1}^{(n)}} h_m^{z_m^{(n)}} A^{z_e^{(n)}}$ $\mathcal{B}$ obtains $(U_n, \mathsf{ch}, z_{m-1}^{(n)}, z_m^{(n)}, z_e^{(n)})$.

- At the end of Step 3 $\mathcal{B}$ can compute the value $U = \prod_{i \in [n]} U_i$ because it knows how to open the commitments to $U_i, i \in [n-1]$ broadcasted by $\mathcal{A}$ (since it is simulating the random oracle).
- $\mathcal{B}$ programs the random oracle setting $\mathcal{H}_{\mathsf{sig}}(\mathsf{nonce}, U, \overline{A}, \overline{B}, \{a_i\}_{i \in \mathsf{Rev}}) = \mathsf{ch}$ so that it can execute the following steps since it knows the responses associated to the first message U_n and the challenge ch.

The simulation of the protocol fails with negligible probability using a similar argument as the one showed in Appendix G.1.

6. At the end of the training, the adversary $\mathcal{A}$ outputs a bit b' specifying their guess about the attribute included in the credential, and $\mathcal{B}$ forwards b' to $\mathcal{C}$.
7. $\mathcal{B}$ forwards b' to $\mathcal{C}$ and wins the hiding experiment with the same advantage of $\mathcal{A}$.

E Identifiable Abort of BBS MHAC Scheme

We prove Lemma 5.

Proof. To prove that CredPres (Protocol 3), satisfies the definition of presentation with identifiable abort, we observe that Protocol 3 already describes the composition of a presentation protocol with the associated identifiable abort detector algorithm W that manifests itself in Step 5 and Step 7.

W takes as input the public inputs to CredPres and processes each message received by a participant P_i. If at any point during the execution of CredPres a message would result in P_i outputting (abort, j) (which might happen only in Step 5 and Step 7), then W aborts with output (abort, j). Otherwise, at the conclusion of the protocol, W outputs $\perp$.

Note that we assume $\mathcal{H}_{\mathsf{com}}$ is a secure commitment scheme, so Step 4 can only be opened to the value committed to in Step 3. Otherwise, by binding, parties would abort in Step 5 identifying the malicious party whose commitment was opened dishonestly. Therefore, we only need to consider when the commitment in Step 3 is opened to the previously committed value by the protocol participants.

We will now argue correctness and identifiability in two parts.

Correctness. Correctness states that whenever the output of W is $\perp$, the output pres of CredPres verifies. That is, if the protocol does not abort, then the output of the protocol verifies. Let us consider a run of the protocol that does not abort:

- P_i does not abort in Step 5: it means that every participant has opened its commitment correctly and it is possible to compute the aggregated first message of the underlying sigma protocol U and therefore the challenge ch.
- P_i does not abort in Step 7: then every $\mathsf{P}_i, i \in S \setminus \{j\}$ has proven knowledge of a representation of $\tilde{B}_i = D_i^r$ w.r.t. $\{h_k\}_{k \in \mathsf{Prv}}, \overline{A}$, and the primary P_j has proven knowledge of $\tilde{B}_j$ with respect to the bases $(C(\mathbf{a}'), \{h_k\}_{k \in \mathsf{Hid}}, \overline{A})$ using the same challenge ch. This means that the proofs of knowledge can be aggregated (see Step 8) resulting in the generation of a valid proof of knowledge of a representation of $\overline{B} = \prod_{k \in S} \overline{B}_k = \prod_{k \in S} \tilde{B}_k$ w.r.t. the bases $(C(\mathbf{a}'), \{h_k\}_{k \in \mathsf{Hid}}, \overline{A})$.

This means that the participants have generated a valid proof of knowledge of a representation of $\overline{B}$ w.r.t. $C(\mathbf{a}'), \{h_k\}_{k\in\mathsf{Hid}}, \overline{A}$, where $\overline{A}$ and $\overline{B}$ satisfy $\mathbf{e}(\overline{A}, X_2) = \mathbf{e}(\overline{B}, g_2)$, therefore the presentation is valid.

Identifiability. Now we consider identifiability, which states that whenever the output is (abort, j) for some $j \in S$, P_j is corrupt. Let us consider the scenarios in which an abort may occur:

- There exists j s.t. P_i outputs (abort, k) in Step 5. It means that P_k did not open correctly its commitment, therefore it is malicious.
- There exists j s.t. P_i outputs (abort, j) in Step 7. It means that P_k did not create a valid coordinated proof of knowledge of a representation of $\tilde{B}_k$ therefore it misbehaved in Step 6 and is identified as corrupt.

F Unforgeability of the BBS MHAC Scheme

We now translate Experiment 3 for the specific case of the BBS multi-holder anonymous credential scheme described in Section 6, then we prove its security.

F.1 Unforgeability experiment instantiation

$\mathsf{Exp}^{\mathsf{c-uf-pres}}_{\mathcal{A}}(\kappa)$:

$\mathsf{pp} \xleftarrow{\$} \mathsf{Pgen}(\kappa)$

$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}(\mathsf{pp})$

$(\mathsf{nonce}^\star, \{a_i^\star\}_{i\in\mathsf{Rev}^\star}, \mathsf{pres}^\star) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{iss}}, \mathcal{O}_{\mathsf{pres}}}(\mathsf{pp}, \mathsf{pk})$

// verification of the validity of the forgery

$\mathtt{flag} \leftarrow 1$

if $\Big((\mathsf{nonce}^\star, \{a_i^\star\}_{i\in\mathsf{Rev}^\star}) \in \mathsf{PT}\Big)$

  $\mathtt{flag} \leftarrow 0$

for $\mathsf{cred} \in \mathsf{CT}$ // for attributes $\{a_i\}_{i\in[m]}$

  if $\{a_i\}_{i\in\mathsf{Rev}^\star} = \{a_i^\star\}_{i\in\mathsf{Rev}^\star}$

    $\mathtt{flag} \leftarrow 0$

return $\Big(\mathsf{VfPres}(\mathsf{nonce}^\star, \{a_i^\star\}_{i\in\mathsf{Rev}}, \mathsf{pres}^\star) = 1\Big) \wedge \mathtt{flag}$

Fig. 5. Description of the multi-holder anonymous credential concurrent unforgeability of presentation experiment $\mathsf{Exp}^{\mathsf{c-uf-pres}}_{\mathcal{A}}(\kappa)$.

Experiment 4 ($\mathsf{Exp}^{\mathsf{c-uf-pres}}_{\mathcal{F},\mathsf{BBS}}$ — Concurrent unforgeability for BBS MHAC scheme).

Setup phase. The challenger of the experiment generates the BBS *public parameters* $\mathsf{pp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2, h_1, \ldots, h_m) \xleftarrow{\$} \mathsf{Pgen}_{\mathsf{BBS}}(\kappa)$ *and the issuer's key pair* $(\mathsf{sk}, \mathsf{pk}) = (x, X_2) \xleftarrow{\$} \mathsf{KGen}_{\mathsf{BBS}}(\mathsf{pp})$ *where* $X_2 = g_2^x$.

Training phase. In our MHAC scheme the adversary needs to access two random oracles, namely $\mathcal{RO}_{\mathsf{com}}, \mathcal{RO}_{\mathsf{sig}}$ *to generate its commitments and the challenge for the presentation computation. The interactions with* $\mathcal{RO}_{\mathsf{com}}, \mathcal{RO}_{\mathsf{sig}}, \mathcal{O}_{\mathsf{iss}}$ *and* $\mathcal{O}_{\mathsf{pres}}$ *are described as follows*

- Random oracle queries*: the adversary has access to a random oracles* $\mathcal{RO}_{\mathsf{sig}}, \mathcal{RO}_{\mathsf{com}}$ *used to create the challenge* ch *and the commitments* com_i *respectively. The adversary can send at most* q_H *queries to the random oracles, where* q_H *is polynomial in the security parameter* κ.
- Credential issuance queries*: the oracle* $\mathcal{O}_{\mathsf{iss}}$ *initialises an empty credential table* CT. *We describe the two kind of queries, namely queries for a full credentials or for the partial credentials. In both cases the adversary always sends queries for credentials without private attributes as we have argued in Observation 1:*
  - *queries for full credentials:* $\mathcal{O}_{\mathsf{iss}}$ *takes in input from* $\mathcal{F}$ *a set of attributes* $\{a_i\}_{i\in[m]}$ *and provides* $\mathcal{F}$ *with a credential* $\mathsf{cred} = ((A, e), \{a_i\}_{i\in[m]})$, *where* (A, e) *is a* BBS *signature on* $\{a_i\}_{i\in[m]}$.[23]
    Finally the oracle stores in CT *a record*

$$(A, e, \{a_i\}_{i\in[m]});$$

  - *query for the target credential:* $\mathcal{F}$ *sends as input to* $\mathcal{O}_{\mathsf{iss}}$ *a tuple*

$$(\{a_i\}_{i\in[m]}, t, n, \mathsf{cor}),$$

    then $\mathcal{O}_{\mathsf{iss}}$ *generates a credential* $\mathsf{cred} = (A, e, \{a_i\}_{i\in[m]})$ *for the set of attributes it was queried and split it in shares according to the parameters* (t, n) *it has received in input generating the shares*

$$\mathsf{cred}_i = (A, e^{(i)}, \{D_i\}_{i\in[n]\setminus\{i\}}, (\{a_j\}_{j\in[m]})).$$

    $\mathcal{F}$ *receives from* $\mathcal{O}_{\mathsf{iss}}$ *the* $|\mathsf{cor}| < t$ *shares of the credential associated to the parties in* cor.
    Finally $\mathcal{O}_{\mathsf{iss}}$ *sets* $\mathsf{targetCred}$ *to*

$$(A, \{e^{(i)}\}_{i\in[n]}, \{D_i\}_{i\in[n]}, \{a_j\}_{j\in[m]}, n, t, \mathsf{cor}).$$

  The number of issuance queries for full credentials $\mathcal{F}$ *can perform is* q_I *which is polynomial in the security parameter* κ.

[23] Note that $\mathcal{F}$ from a BBS signature can generate a multi-holder BBS credential for any possible parameter (t, n) choice.

– Presentation queries*: the presentation oracle* $\mathcal{O}_{\mathsf{pres}}$ *initialises the presentation table* PT *and has reading access to record* targetCred*. The adversary can query the presentation oracle* $\mathcal{O}_{\mathsf{pres}}$ *for presentations of the target credential.* Query for the presentation of targetCred: $\mathcal{F}$ *inputs to* $\mathcal{O}_{\mathsf{pres}}$ *a tuple* $(\mathsf{nonce}, \{a_i\}_{i\in\mathsf{Rev}}, \mathsf{hon})$ *where* $\{a_i\}_{i\in\mathsf{Rev}}$ *are a subset of the attributes included in* targetCred*.* $\mathcal{O}_{\mathsf{pres}}$ *interacts with* $\mathcal{F}$ *in the creation of a presentation on behalf of the parties in* hon *and at the end stores in* PT *the record* $(\mathsf{nonce}, \{a_i\}_{i\in\mathsf{Rev}})$*. The adversary possibly opens many concurrent sessions of presentation of the target credential, each of them is identified by an identifier* ssid*.*

Forgery phase. At the end, $\mathcal{F}$ *must perform its forgery and sends the challenger a presentation* $\mathsf{pres}^\star = (\overline{A}, \overline{B}, \mathsf{ch}, z_r, \{z_j\}_{j\in\mathsf{Hid}}, z_e)$ *for the statement* $\{a_i^\star\}_{i\in\mathsf{Rev}^\star}$ *and nonce* $\mathsf{nonce}^\star$*.*

Winning conditions. The winning conditions are the same as described in Experiment 3.

According to Definition 12, a protocol for the presentation of a BBS multi-holder anonymous credentials is unforgeable if $\Pr\left[\mathsf{Exp}^{\mathsf{c-uf-pres}}_{\mathcal{F},\mathsf{BBS}}(\kappa) = 1\right]$ is negligible in the security parameter κ.

F.2 Unforgeability proof

In this section we prove the unforgeability of the multi-holder BBS presentation protocol described in Protocol 3.

Proof. The proof is divided in two cases, corresponding to the two ways the adversary can win the experiment: in the first (Case A) we consider an adversary who wins the experiment producing a forgery associated to the target credential issued to it during the experiment, in the second (Case B) we consider an adversary who produces a forgery for a credential it has never been issued.

Standard reduction simplifications. In the security proof we will apply the following simplification to our model:

– *From* (t, n)*-Shamir secret sharing to additive* (n, n) *secret sharing.* For the sake of simplicity, and without loss of generality, as we did for the description of Protocol 3, in the security proof we assume that the issuance of the partial credential happens using a full threshold (n, n)-secret sharing (additive secret sharing). We will also assume that $\mathsf{cor} = [n-1]$, therefore the adversary will be provided with the $n-1$ shares

$$(A, \{e^{(i)}\}_{i\in[n-1]}, \{D_i\}_{i\in[n]}, \{a_j\}_{j\in[m]})$$

where $e = \sum_{i=1}^{n} e^{(i)}$, $D \leftarrow A^e$ and the issue oracle $\mathcal{O}_{\mathsf{iss}}$ passes to the presentation oracle $\mathcal{O}_{\mathsf{pres}}$ the record targetCred equal to

$$(A, \{e^{(i)}\}_{i\in[n]}, \{a_j\}_{j\in[m]}, n, n, [n-1]).$$

– *A mandatory random oracle query.* We will assume that $\mathcal{F}$, before sending to $\mathcal{B}$ its forgery of presentation $(\overline{A}, \overline{B}, \mathsf{ch}, z_r, \{z_j\}_{j\in\{\mathsf{Hid}\}}, z_e)$ for the message nonce and revealed attributes $\{a_i\}_{i\in\mathsf{Rev}}$, has queried the random oracle $\mathcal{RO}_{\mathsf{sig}}$ on the input

$$(\mathsf{ssid}, \mathsf{nonce}, U, \overline{A}, \overline{B}, \{a_j\}_{j\in\mathsf{Rev}}).$$

Otherwise one can program a forger $\mathcal{F}'$ who does the same operations of $\mathcal{F}$ but before outputting the forgery sends this extra query to $\mathcal{RO}_{\mathsf{sig}}$ to satisfy this assumption. In such case we must increase the maximum amount of queries $\mathcal{F}$ can perform to $q_H + 1$.

Case A: reduction to DL Reduction $\mathcal{B}$ simulates the challenger of $\mathsf{Exp}^{\mathsf{c-uf-pres}}_{\mathcal{F},\mathsf{BBS}}(\kappa)$ while interacting with $\mathcal{F}$. Let us show how $\mathcal{B}$ simulates the public data generation, the oracles $\mathcal{RO}_{\mathsf{com}}, \mathcal{RO}_{\mathsf{sig}}, \mathcal{O}_{\mathsf{iss}}$ and $\mathcal{O}_{\mathsf{pres}}$, and finally how it uses $\mathcal{F}$ to solve its challenge in $\mathsf{Exp}^{\mathsf{DL}}_{\mathcal{B}}(\kappa)$ which takes place in the group $\mathbb{G}_1$, the same used for BBS.

In this reduction we consider an adversary $\mathcal{F}$ who wins by forging a presentation using the target credential it is issued.

Setup, public parameters and key generation. The reduction $\mathcal{B}$ receives in input the tuple $(p, \mathbb{G}_1, g, h)$ from $\mathcal{C}_{\mathsf{DL}}$, where $(g, h) \in \mathbb{G}_1^2$ is an instance of the discrete logarithm problem that $\mathcal{B}$ needs to solve. $\mathcal{B}$ generates the public parameters and the issuer's key pair for the BBS signature scheme as described in Figure 6. The simulation of the parameter generation and key generation is indistinguishable from a real execution of the parameter generation because the key generation is computed in the exact same way, and the elements $(g_2, g_1, h_1, \ldots, h_m)$ are chosen uniformly at random in $\mathbb{G}_2 \times \mathbb{G}_1^{m+1}$. However $\mathcal{B}$ knows the discrete logarithm of the elements in G_1 with respect to $k = g^x h$.

Random oracles simulation. $\mathcal{B}$ can simulate the random oracles $\mathcal{RO}_{\mathsf{com}}$ and $\mathcal{RO}_{\mathsf{sig}}$ programming them using the hash tables $\mathsf{HT}_{\mathsf{com}}, \mathsf{HT}_{\mathsf{sig}}$ which are initialised to empty and whenever one of the two oracles is queried a message m, $\mathcal{B}$ checks the corresponding hash table and, if there is an entry (m, d) (in such case we write $\mathsf{HT}(m) = d$), returns the digest d, otherwise it samples $d \xleftarrow{\$} \mathbb{Z}_p$, stores in the corresponding hash table the record (m, d) and returns d.

The counter c_H used in the description of the simulation of the random oracles in Figure 7 is shared among the two interfaces $\mathcal{B}.\mathcal{RO}_{\mathsf{sig}}$ and $\mathcal{B}.\mathcal{RO}_{\mathsf{com}}$.

Issuing oracle simulation. We distinguish two cases corresponding to the issuance queries for full credentials and the single query for the target credential.

– When $\mathcal{F}$ sends an issuance query for a full credential for the attributes $\{a_i\}_{i\in[m]}$, $\mathcal{B}$ simulates $\mathcal{O}_{\mathsf{iss}}$ by generating a credential according to the BBS signing algorithm $\mathsf{Sign}_{\mathsf{BBS}}(x, \{a_i\}_{i\in[m]}) \xrightarrow{\$} (A, e)$, and sending it to $\mathcal{F}$. Then

$\mathcal{B}.\mathsf{Setup}(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g, h)$

$g_2 \xleftarrow{\$} \mathbb{G}_2$ // generator for $\mathbb{G}_2$

$x \xleftarrow{\$} \mathbb{Z}_p$ // BBS issuer secret key

$X_2 \leftarrow g_2^x$ // BBS issuer public key

$k \leftarrow g^x h$ // note that if $\log_g h = e$, then $g^x h = g^{x+e}$

$\gamma_0, \gamma_1, \ldots, \gamma_m \xleftarrow{\$} \mathbb{Z}_p$

$g_1 \leftarrow k^{\gamma_0}$ // generator for $\mathbb{G}_1$

$h_i = k^{\gamma_i}, \forall i \in [m]$

$\mathsf{pp} \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2, h_1, \ldots, h_m)$

$\mathsf{HT}_{\mathsf{com}}, \mathsf{HT}_{\mathsf{sig}} \leftarrow \emptyset$ // hash tables

$\mathsf{CT}, \mathsf{PT} \leftarrow \emptyset$ // credential and presentation tables

$c_H, c_I, c_P \leftarrow 0$ // counters for random oracle, issuing and presentation queries

$\mathsf{targetCred} \leftarrow \perp$ // the queried partial credential

$(\mathsf{pp}, X_2) \twoheadrightarrow \mathcal{F}$

return $(\mathsf{pp}, X_2, \mathsf{HT}_{\mathsf{com}}, \mathsf{HT}_{\mathsf{sig}}, \mathsf{CT}, \mathsf{PT}, c_H, c_I, c_P)$

Fig. 6. Simulation of the setup algorithm of the unforgeability experiment

$\mathcal{B}.\mathcal{RO}_{\mathsf{com}}(m)$	$\mathcal{B}.\mathcal{RO}_{\mathsf{sig}}(m)$
if $c_H < q_H \wedge \mathsf{HT}_{\mathsf{com}}(m) = \perp$	**if** $c_H < q_H \wedge \mathsf{HT}_{\mathsf{sig}}(m) = \perp$
$c_H \leftarrow c_H + 1$	$c_H \leftarrow c_H + 1$
$d \xleftarrow{\$} \mathbb{Z}_p$	$d \xleftarrow{\$} \mathbb{Z}_p$
$\mathsf{HT}_{\mathsf{com}}.\mathsf{add}(m, d)$	$\mathsf{HT}_{\mathsf{sig}}.\mathsf{add}(m, d)$
return $\mathsf{HT}_{\mathsf{com}}(m)$	**return** $\mathsf{HT}_{\mathsf{sig}}(m)$

Fig. 7. Simulation of the random oracles of the unforgeability experiments (Case A).

$\mathcal{B}$ stores in CT the credential $\mathsf{cred} \leftarrow (A, e, \{a_i\}_{i\in[m]})$. In this case the simulation is perfect.

– When $\mathcal{F}$ sends its only issuance query for the target credential giving in input $(\{a_i\}_{i\in[m]}, t, n, \mathsf{cor})$, $\mathcal{B}$ simulates the issuance of the target credential so that the value $e^\star$ of the underlying BBS signature $(A, e^\star)$ is actually the discrete logarithm $\log_g h$[24]. To do so, $\mathcal{B}$ must produce two values A, D s.t. $\log_A D = \log_g h$, and to do so, it exploits its knowledge of the discrete logarithms of $g_1, h_1, \ldots, h_m$ w.r.t. k, and the fact that $k = g^x h$ as we show in Figure 8. We recall that for the sake of simplicity we assume that $t = n$ and that $\mathcal{F}$ always chooses to corrupt the parties in $\mathsf{cor} = [n-1]$.
The simulation of the issuance of the target credential is perfect as well. The resulting value D will satisfy $D = A^{e^\star}$, where $e^\star$ is the solution of the input challenge to $\mathcal{B}$, i.e. $e^\star = \log_g h$. Moreover, the set $\{e^{(i)}\}_{i\in\mathsf{cor}}$ is indistinguishable from an honest execution since, being $|\mathsf{cor}| < t$ they are selected uniformly at random.

Presentation oracle simulation. When $\mathcal{F}$ sends a presentation query for the target credential, $\mathcal{F}$ inputs to $\mathcal{O}_{\mathsf{pres}}$ a tuple $(\mathsf{nonce}, \{a_i\}_{i\in\mathsf{Rev}}, \mathsf{hon})$.

$\mathcal{B}$ simulates $\mathcal{O}_{\mathsf{pres}}$ by retrieving the tuple $\mathsf{targetCred}$ equal to

$$(A, \{e^{(i)}\}_{i\in\mathsf{cor}}, \{D_i\}_{i\in[n]}, \{a_j\}_{j\in[m]}, n, t, \mathsf{cor})$$

where $\mathcal{B}$ does not know the values in $\{e^{(i)}\}_{[n]\setminus\mathsf{cor}} (= e^{(n)})$ which are set to $\perp$ since $e^\star = \log_g h$ is unknown to $\mathcal{B}$.

$\mathcal{B}$ must simulate an interaction between the parties in cor controlled by $\mathcal{F}$ and the parties in hon controlled by $\mathcal{O}_{\mathsf{pres}}$. Without loss of generality we can assume that the primary party is one of the corrupted participants, so that the adversary can choose the value r in the first step. The case where the primary party is controlled by $\mathcal{B}$ can be simulated likewise.

Since we are adopting the simplification of additive secret sharing, we recall that $t = n$ so we do not consider the Lagrange coefficients in the aggregation of the shares of e.

The simulation of the execution of Protocol 3 using as nonce nonce and revealing the attributes $\{a_i\}_{i\in\mathsf{Rev}}$ works as follows:

$\mathcal{F}$ broadcasts, on behalf of the primary party $\mathsf{P}_j, j \in \mathsf{cor}$, the value r used to generate $\overline{A}$ and $\overline{B}$.

Every party can compute the values $\overline{A}, C(\mathbf{a}'), \overline{B}$ as prescribed by the presentation protocol (Protocol 3) in Step 2.

Then the participants must generate the proof of knowledge of a representation of $\overline{B}$ w.r.t. $C(\mathbf{a}'), \{h_i\}_{i\in\mathsf{Hid}}, \overline{A}$ and this happens by having each party generate a proof of knowledge of:

[24] Note that $\mathcal{F}$ in this experiment knows all the attributes since it can choose them, even the secret shared ones, but we will show that it will not be able to forge a presentation

$\mathcal{B}.\mathsf{IssueCred}(\mathbf{a} = \{a_i\}_{i\in[m]}, t, n, \mathsf{cor})$

if $(t, n, \mathsf{cor}) = (\bot, \bot, \bot) \wedge (c_I < q_I)$ // full credential query

$c_I \leftarrow c_I + 1$

$(A, e) \xleftarrow{\$} \mathsf{Sign}_{\mathsf{BBS}}(\mathsf{sk} = x, \{a_i\}_{i\in[m]})$

$\mathsf{cred} \leftarrow ((A, e), \{a_i\}_{i\in[m]})$

$\mathsf{CT}.\mathsf{add}(\mathsf{cred})$

return cred

if $(t, n, \mathsf{cor}) \neq (\bot, \bot, \bot) \wedge \mathsf{targetCred} = \bot$ // partial credential query

$\alpha \leftarrow \gamma_0 \sum_{i=1}^{m} \gamma_i a_i$

$A \leftarrow g^{\alpha}$ // if $h = g^{e^\star}$ then $k = g^{x+e^\star}$ i.e. $A = (k^{\alpha})^{\frac{1}{x+e^\star}} = (C(\mathbf{a}))^{\frac{1}{x+e^\star}}$

$D \leftarrow (h^{\alpha})^{-1}$ // D would be $A^{-e^\star}$

// Note that $A^x D = A^{x+e^\star} = A^x A^{e^\star} = g^{\alpha x} h^{\alpha} = (g^x h)^{\alpha} = k^{\alpha} = g_1 \prod_{i=1}^{m} h_i^{a_i}$

$\{e^{(i)}\}_{i\in\mathsf{cor}} \xleftarrow{\$} \mathbb{Z}_p^{|\mathsf{cor}|}$

$D_i \leftarrow A^{-e^{(i)}}, \forall i \in \mathsf{cor}$ // we assume $\mathsf{cor} = [n-1]$

$D_n \leftarrow D \prod_{i\in\mathsf{cor}} D_i^{-1}$ // $= A^{-e^\star} \prod_{i\in\mathsf{cor}} A^{e^{(i)}}$

$\{\mathsf{cred}_i\}_{i\in\mathsf{cor}} \leftarrow ((A, \{e^{(i)}\}_{i\in\mathsf{cor}}, \{D_i\}_{i\in[n]}, \{a_j\}_{j\in[m]})$

$\mathsf{targetCred} \leftarrow ((A, \{e^{(i)}\}_{i\in\mathsf{cor}}, \{D_i\}_{i\in[n]}, \{a_j\}_{j\in[m]}, t, n, \mathsf{cor})$

// Note that $e^{(n)}$ is not known to $\mathcal{B}$

return $\{\mathsf{cred}_i\}_{i\in[\mathsf{cor}]}$

return $\bot$

Fig. 8. Simulation of the answers to issuance queries in the unforgeability experiment (Case A).

– $\tilde{B}_i = D_i^r$ w.r.t. the basis $\overline{A}$ for each $i \neq j$;
– whereas the primary participant P_j will generate a proof of knowledge of $\tilde{B}_j = C(a')^r D_j^r \prod_{k \in \mathsf{Hid}} h_k^{ra_k}$ w.r.t. the bases $C(\mathbf{a}'), \{h_j\}_{j \in \mathsf{Hid}}, \overline{A}$.

Recall that every party can compute the values $\tilde{B}_i, i \in [n]$ from the known values D_i and r, and also that the reduction $\mathcal{B}$, acting on behalf of P_n does not know the value $e^{(n)}$ but can compute the value $\tilde{B}_n$, as it knows the value $A^{e^{(n)}}$.

At this point, the reduction $\mathcal{B}$ simulates the execution of the sigma protocol to prove knowledge of the representation of $\tilde{B}_n$.

Since $\mathcal{B}$ does not know the value $e^{(n)}$, it must simulate the protocol execution by firstly computing its responses $z_e^{(n)} \xleftarrow{\$} \mathbb{Z}_p$ and a challenge $\mathsf{ch} \xleftarrow{\$} \mathbb{Z}_p$. Then $\mathcal{B}$ sets $U_n \leftarrow \tilde{B}_n^{-\mathsf{ch}} \overline{A}^{z_e^{(n)}}$.

$\mathcal{B}$ sets $\mathsf{com}_n \leftarrow \mathcal{H}_{\mathsf{com}}(\mathsf{ssid}, \mathsf{nonce}, U_n)$ if $(\mathsf{ssid}, \mathsf{nonce}, U_n)$ was previously included in the hash table $\mathsf{HT}_{\mathsf{com}}$, otherwise samples $\mathsf{com}_n \xleftarrow{\$} \mathbb{Z}_p$ and programs $\mathsf{HT}_{\mathsf{com}}$ adding the entry $(\mathsf{com}_n, (\mathsf{nonce}, U_n))$. $\mathcal{B}$ broadcasts com_n to the other participants.

When $\mathcal{B}$ has received the values com_i from the other participants, it must have received an hash query for (nonce, U_i) such that there is an entry $(\mathsf{com}_i, (\mathsf{ssid}, \mathsf{nonce}, U_i))$ in $\mathsf{HT}_{\mathsf{com}}$, therefore $\mathcal{B}$ can compute in advance the value $U \leftarrow \prod_{i \in [n]} U_i$ and program the random oracle $\mathcal{H}_{\mathsf{sig}}$ adding to the hash table $\mathsf{HT}_{\mathsf{sig}}$ an entry $(\mathsf{ch}, (\mathsf{ssid}, \mathsf{nonce}, U, \overline{A}, \overline{B}, \{a_i\}_{i \in \mathsf{Rev}}))$.

Event bad. The simulation might fail if the random oracle $\mathcal{H}_{\mathsf{sig}}$ needs to be overwritten and we call this event bad.

Given that the challenge associated to the protocol execution is the one corresponding to the simulated transcript, $\mathcal{B}$ does not have to compute the response and can return the values sampled at random at the beginning of the simulation, namely $z_e^{(n)}$.

This allows $\mathcal{B}$ to simulate the presentation protocol execution, and also to perform all the identifiable abort checks.

This terminates the simulated execution of the presentation query of a partial credential. The probability that $\mathcal{B}$ correctly simulates the experiment is

$$\Pr[\mathcal{B} \text{ simulates}] \geq 1 - \left(\frac{(q_H + q_P)^2}{p} \right)$$

which is overwhelming in the security parameter and is evaluated in Appendix G.1.

Exploit of the forgery. Eventually the adversary $\mathcal{F}$ outputs a forgery

$$(\mathsf{nonce}, \{a_i\}_{i \in \mathsf{Rev}}, \mathsf{pres})$$

which is valid, i.e. $\mathsf{VfPres}_{\mathsf{BBS}}(\mathsf{pp}, \mathsf{pk}, \mathsf{nonce}, \{a_i\}_{i \in \mathsf{Rev}}, \mathsf{pres}) = 1$, with

$$\mathsf{pres} = \big(\overline{A}, \overline{B}, \mathsf{ch}, (z_r, \{z_j\}_{j \in \mathsf{Hid}}, z_e)\big).$$

Being $U \leftarrow \overline{B}^{-\mathsf{ch}} C(\mathbf{a}')^{z_r} \prod_{j \in \mathsf{Hid}} h_j^{z_j} \overline{A}^{z_e}$, according to our reduction simplifications, $\mathcal{F}$ must have sent a query to $\mathcal{RO}_{\mathsf{sig}}$ for $(\mathsf{ssid}, \mathsf{nonce}, \overline{A}, \overline{B}, U, \{a_k\}_{k \in \mathsf{Rev}})$, which returned the value ch and this query happens after the signature material randomization phase, since $\overline{A}, \overline{B}$ must be determined.

$\mathcal{B}$ rewinds $\mathcal{F}$ to the moment in which it performed such query (which is after the signature randomization phase) and sets $\mathsf{HT}_{\mathsf{sig}}(\{a_k\}_{k \in \mathsf{Rev}}, \mathsf{nonce}, \overline{A}, \overline{B}, U) \leftarrow \mathsf{ch}'$.

By the general forking lemma [BN06] (a formalisation of the Forking Lemma in [PS96]), with non-negligible probability the adversary $\mathcal{F}$ will end the experiment execution outputting a forgery associated to the same hash query (therefore also for the same $(\mathsf{nonce}, \{a_i\}_{i \in \mathsf{Rev}}, \mathsf{pres}')$) such that

$$\mathsf{pres}' = \left(\overline{A}, \overline{B}, \mathsf{ch}', (z_r', \{z_j'\}_{j \in \mathsf{Hid}}, z_e')\right)$$

and

$$\overline{B}^{-\mathsf{ch}'} C(\mathbf{a}')^{z_r'} \Big(\prod_{j \in \mathsf{Hid}} h_j^{z_j'} \Big) \overline{A}^{z_e'} = U.$$

This allows us to extract the credential $((A, e^\star), \{a_i\}_{i \in [m]})$ using the algorithm shown in [TZ23] and that we recall in Appendix B.

We evaluate the advantage $\mathsf{Adv}^{\mathsf{DL}}_{\mathcal{B}}(\kappa)$ of the reduction in winning the DL experiment as a function of the advantage $\mathsf{Adv}^{\mathsf{c-uf-pres}_{(A)}}_{\mathcal{F},\mathsf{BBS}}(\kappa)$ of the adversary $\mathcal{F}$ of the presentation unforgeability experiment in Appendix G.1 where we show that:

$$\mathsf{Adv}^{\mathsf{DL}}_{\mathcal{B}}(\kappa) \geq \frac{\left(1 - \frac{q^2}{p}\right)^2}{q_H} \left(\mathsf{Adv}^{\mathsf{c-uf-pres}_{(A)}}_{\mathcal{F},\mathsf{BBS}}(\kappa)\right)^2 - \frac{\left(1 - \frac{q^2}{p}\right)}{p} \left(\mathsf{Adv}^{\mathsf{c-uf-pres}_{(A)}}_{\mathcal{F},\mathsf{BBS}}(\kappa)\right). \quad (6)$$

Therefore if there exists an adversary which wins with non-negligible probability the unforgeability experiment we could design an adversary of the DL experiment which wins with probability which is non-negligible as we have shown above.

Since in this section we consider the case in which the extracted credential is $\mathsf{targetCred}$, $\mathcal{B}$ extracts $(A, e^\star), \{a_1, \ldots, a_m\}$ such that $A^{e^\star} = D$ according to the notation used in the description of the simulation of $\mathcal{O}_{\mathsf{iss}}$. Since $A = g^\alpha$ and $D = h^\alpha$, then we can state that $h = g^{e^\star}$, therefore $\mathcal{B}$ sends to the challenger of the discrete logarithm experiment the value e winning the experiment every time that $\mathcal{F}$ wins.

Case B: reduction to $\mathsf{suf-cma}$ for BBS. Tessaro and Zhu in [TZ23] prove the BBS signature scheme strongly unforgeable against chosen message attacks under the qSDH assumption. In this section we prove that an attacker $\mathcal{F}$ of

$\mathsf{Exp}^{\mathsf{c-uf-pres}}_{\mathcal{F},\mathsf{BBS}}(\kappa)$ can be used to design an attacker $\mathcal{B}$ of the strong unforgeability experiment for the BBS signature scheme $\mathsf{Exp}^{\mathsf{suf-cma}}_{\mathcal{B},\mathsf{BBS}}(\kappa)$ described in [TZ23, Figure 1].

A corollary of our proof is that the unforgeability of presentations of the multi-holder BBS anonymous credential scheme reduces to the qSDH assumption.

In this reduction we consider an adversary $\mathcal{F}$ which forges a presentation associated to a credential it has never been issued.

Setup, public parameters and key generation $\mathcal{B}$ receives the BBS public parameters from the challenger of the experiment $\mathsf{Exp}^{\mathsf{suf-cma}}_{\mathcal{F},\mathsf{BBS}}(\kappa)$. The reduction $\mathcal{B}$ performs the following operations:

$\mathcal{B}.\mathsf{Setup}(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2, h_1, \ldots, h_m, X_2)$

$\mathsf{pp} \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2, h_1, \ldots, h_m)$
$\mathsf{pk} \leftarrow X_2$
$\mathsf{HT}_{\mathsf{com}}, \mathsf{HT}_{\mathsf{sig}} \leftarrow \emptyset$ // hash tables
$\mathsf{CT}, \mathsf{PT} \leftarrow \emptyset$ // credential and presentation tables
$c_H, c_I, c_P \leftarrow 0$ // counters for random oracle, issuing and presentation queries
$\mathsf{targetCred} \leftarrow \perp$ // the queried partial credential
$(\mathsf{pp}, X_2) \twoheadrightarrow \mathcal{F}$
return $(\mathsf{pp}, X_2, \mathsf{HT}_{\mathsf{com}}, \mathsf{HT}_{\mathsf{sig}}, \mathsf{CT}, \mathsf{PT}, c_H, c_I, c_P)$

Random oracles simulation. The random oracles are simulated as described in Case A.

Issuing oracle simulation. When the reduction $\mathcal{B}$ receives a issuing query, with input $(\{a_i\}_{i\in[m]}, t, n, \mathsf{cor})$ it sends a signing query to the oracle $\mathcal{O}_{\mathsf{SIGN}}$ of the strong unforgeability experiment for BBS signatures for the messages $\{a_i\}_{i\in[m]}$ receiving a signature (A, e) on these messages. Then if it was a full credential query $\mathcal{B}$ sends $((A, e), \{a_i\}_{i\in[m]})$ to $\mathcal{F}$ and stores the credential in the credential table CT. If it is the single partial credential query $\mathcal{F}$ can perform, $\mathcal{B}$ generates a (t, n) secret sharing of $e, \{e^{(i)}\}_{i\in[n]} \xleftarrow{\$} \mathsf{Share}(t, n, e)$ and sets targetCred to be equal to

$$(A, \{e^{(i)}\}_{i\in[n]}, \{D_i\}_{i\in[n]}, \{a_j\}_{j\in[m]}, t, n, \mathsf{cor})$$

and sends to $\mathcal{F}$ the shares of credentials corresponding to the parties in cor.

$\mathcal{B}.\mathsf{IssueCred}(\{a_i\}_{i\in[m]}, t, n, \mathsf{cor})$

if $(t, n, \mathsf{cor}) = (\bot, \bot, \bot) \wedge c_I < q_I$ // full credential query
  $c_I \leftarrow c_I + 1$
  $\{a_i\}_{i\in[m]} \rightarrowtriangle \mathcal{O}_{\mathsf{SIGN}}$ // send a sign query to the BBS signing oracle.
  $(A, e) \leftarrowtriangle \mathcal{O}_{\mathsf{SIGN}}$
  $\mathsf{cred} \leftarrow ((A, e), \{a_i\}_{i\in[m]})$
  $\mathsf{CT}.\mathsf{add}(\mathsf{cred})$
  return cred
if $(t, n, \mathsf{cor}) \neq (\bot, \bot, \bot) \wedge \mathsf{targetCred} = \bot$ // partial credential query
  $\{a_i\}_{i\in m} \rightarrowtriangle \mathcal{O}_{\mathsf{SIGN}}$ // send a sign query to the BBS signing oracle.
  $(A, e) \leftarrowtriangle \mathcal{O}_{\mathsf{SIGN}}$
  $\{e^{(i)}\}_{i\in[n]} \xleftarrow{\$} \mathsf{Share}(t, n, e)$
  $D_i \leftarrow A^{-e^{(i)}}, \forall i \in [n]$
  $\{\mathsf{cred}_i\}_{i\in\mathsf{cor}} \leftarrow (A, \{D_i\}_{i\in[n]}, \{e^{(i)}\}_{i\in\mathsf{cor}}, \{a_k\}_{k\in[m]})$
  // In the reduction we will consider $t = n$ and $\mathsf{cor} = [n-1]$
  $\mathsf{targetCred} \leftarrow (A, \{e^{(i)}\}_{i\in[n]}, \{D_i\}_{i\in[n]}, \{a_j\}_{j\in[m]}, t, n, \mathsf{cor})$
  return $\{\mathsf{cred}_i\}_{i\in[\mathsf{cor}]}$
return $\bot$

Presentation oracle simulation. Regarding the queries for presentations of the target credential, the reduction $\mathcal{B}$ simply executes the presentation protocol since it knows all the shares of the credentials associated to $\mathsf{targetCred}$ and in this case the simulation is perfect since the reduction knows all the information needed to execute the protocol steps.

Exploit of the forgery. The adversary $\mathcal{F}$ eventually outputs a forgery. $\mathcal{B}$ rewinds $\mathcal{F}$ as we have described in Case A, and manages to extract a credential $\mathsf{cred}^\star = ((A, e), \{a_i\}_{i\in[m]})$.

In this section we consider the case in which the extracted credential is associated to a credential never issued to $\mathcal{F}$, therefore $\mathsf{cred}^\star \notin \mathsf{CT}$, and either $\{a_i\}_{i\in[m]}$ was never queried to $\mathcal{O}_{\mathsf{SIGN}}$, the sign oracle of the strong unforgeability experiment, or it was queried but returned a different signature (A', e'), therefore $\mathcal{B}$ can send to the challenger of $\mathsf{Exp}^{\mathsf{suf-cma}}_{\mathcal{F},\mathsf{BBS}}(\kappa)$ the forgery $((A, e), \{a_i\}_{i\in[m]})$, which is a valid forgery for the strong unforgeabilty experiment for the BBS signature.

We evaluate the advantage $\mathsf{Adv}^{\mathsf{suf-cma}}_{\mathcal{B},\mathsf{BBS}}(\kappa)$ of the reduction $\mathcal{B}$ in winning the strong unforgeability experiment for BBS signatures [TZ23] as a function of the advantage $\mathsf{Adv}^{\mathsf{c-uf-pres}_{(B)}}_{\mathcal{F},\mathsf{BBS}}(\kappa)$ of the adversary $\mathcal{F}$ of the presentation unforgeabil-

ity experiment in Appendix G.2 where we show that:

$$\mathsf{Adv}^{\mathsf{suf-cma}}_{\mathcal{B},\mathsf{BBS}}(\kappa) \geq \frac{\left(\mathsf{Adv}^{\mathsf{c-uf-pres}_{(B)}}_{\mathcal{F},\mathsf{BBS}}(\kappa)\right)^2}{q_H} - \frac{\left(\mathsf{Adv}^{\mathsf{c-uf-pres}_{(B)}}_{\mathcal{F},\mathsf{BBS}}(\kappa)\right)}{p}. \quad (7)$$

□

Remark 5. If the MHAC scheme satisfies the unforgeability of presentations, this implies that also the credential issuing algorithm is unforgeable; otherwise, the adversary could forge a credential for a set of attributes different from the ones included in the credentials it was issued and create a valid presentation out of it.

G Advantage of the Presentation Unforgeability Reduction

In this section we provide an upper-bound to the advantage of the reduction in winning the DL experiment (Case A) or the SUF-CMA experiment for BBS signatures (Case B).

We first upper-bound the probability that the reduction $\mathcal{B}$ fails the simulation of the presentation unforgeability experiment, then we find a lower-bound to the advantage of the reduction in winning the DL experiment (Case A) or the strong unforgeabilty under chosen message attack experiment for BBS signatures (Case B).

G.1 Case A: extract the target credential

Simulation failure probability We upper-bound the probability that one of the events bad occurs as a consequence of the queries of $\mathcal{F}$. The events happen if the hash table $\mathsf{HT}_{\mathsf{sig}}$ is overwritten during the simulation.

Being $i \in [q_P]$ the index representing the i-th presentation query performed by $\mathcal{F}$, and by $\mathsf{bad}^{(i)}$ the event that bad happens during the i-th presentation query, we can upper-bound the probability that the event $\mathsf{bad}^{(i)}$ happens.

Note that these events can be represented as the random sampling of an element from a large set of distinct elements which results to assume a value belonging to a small set of values which are the values previously included in the hash tables. Note that the query for

$$(\mathsf{nonce}, U, \overline{A}, \overline{B}\{a_i\}_{i \in \mathsf{Rev}})$$

has each component chosen by $\mathcal{F}$ apart from U which results to be sampled at random. Being q_H the number of hash queries $\mathcal{F}$ can perform and q_P the number of presentation queries, the hash table $\mathsf{HT}_{\mathsf{sig}}$ has at most $q_H + q_P$ elements during the simulation execution, therefore the probability that event $\mathsf{bad}^{(i)}$ happens is less than $\frac{q_H+q_P}{p}$ where p is the number of possible values of U.

Let $q \leftarrow q_H + q_P$, then

$$\Pr[\mathcal{B} \text{ simulates}] = \Pr\left[\bigwedge_{i\in[q_P]} \neg\mathsf{bad}^{(i)}\right] \geq$$
$$\prod_{i\in[q_P]}\left(1-\frac{q}{p}\right) \geq \left(1-\frac{q}{p}\right)^q \geq 1-\left(\frac{q^2}{p}\right) \quad (8)$$

where the last inequality holds because

$$\left(1-\frac{q}{p}\right)^q = 1+\sum_{i\in[q]}\binom{q}{i}\left(-\frac{q}{p}\right)^i$$

and for each j,

$$\binom{q}{j}\left(\frac{q}{p}\right)^j \geq \binom{q}{j+1}\left(\frac{q}{p}\right)^{j+1}$$

since writing explicitly the products, one obtains

$$\frac{1}{q-j} \geq \frac{q}{(j+1)p}$$

which holds for $p \gg q$.

Therefore, assuming that q is odd [25]

$$\sum_{i\in[q]}\binom{q}{i}\left(-\frac{q}{p}\right)^i =$$
$$1-\frac{q^2}{p}+\sum_{i=2,i \text{ even}}^{q-1}\left[\binom{q}{i}\left(-\frac{q}{p}\right)^i-\binom{q}{i+1}\left(\frac{q}{p}\right)^{i+1}\right] \geq 1-\frac{q^2}{p} \quad (9)$$

Note that the simulation takes a polynomial time to be executed and do not requires to rewind the adversary $\mathcal{F}$.

Therefore the simulation is successful with probability

$$\Pr[\mathcal{B} \text{ simulates}] \geq 1-\left(\frac{(q_H+q_P)^2}{p}\right)$$

which is overwhelming in the security parameter κ, being q polynomial in κ and p super-polynomial.

---

[25] If q is even, we should add an extra *positive* element $\left(\frac{2q}{p}\right)^{3q}$ which makes the relation hold anyways.

Advantage of the reduction. We can finally evaluate the advantage of $\mathcal{B}$ in winning the discrete logarithm experiment as a function of the advantage of $\mathcal{F}$ in winning the unforgeability of presentation experiment.

Let us define

- $\mathsf{Adv}^{\mathsf{DL}}_{\mathcal{B}}(\kappa)$ as the probability that $\mathcal{B}$ interacting with $\mathcal{F}$ wins the DL experiment;
- $\mathsf{Adv}^{\mathsf{c-uf-pres}_{(A)}}_{\mathcal{F},\mathsf{BBS}}(\kappa) = \Pr\left[\mathcal{F}^{\mathcal{B}} \text{ wins } \mathsf{c-uf-pres}_{(A)} | \mathcal{B} \text{ simulates}\right]$ as the probability that $\mathcal{F}$ wins the unforgeability of presentation forging the target credential according to case A, which is equal to the probability that $\mathcal{F}$ wins the experiment interacting with $\mathcal{B}$, provided that $\mathcal{B}$ correctly simulates the experiment.

Since $\mathcal{B}$ interacting with $\mathcal{F}$ wins the DL experiment if it correctly simulates the challenger of the unforgeability experiment and $\mathcal{F}$ forges a presentation both before and after being rewound, we can apply the Generalised Forking Lemma [BN06] to the algorithm that is successful if

- $\mathcal{F}$ produces a valid forgery (which happens with probability $\mathsf{Adv}^{\mathsf{c-uf-pres}_{(A)}}_{\mathcal{F}}(\kappa)$);
- $\mathcal{B}$ does not fail the simulation of the experiment.

This algorithm succeeds with probability

$$\begin{aligned}\epsilon_A = \Pr\left[\mathcal{F}^{\mathcal{B}} \text{ wins } \mathsf{c-uf-pres}_{(A)} \wedge \mathcal{B} \text{ simulates}\right] = \\ \Pr[\mathcal{B} \text{ simulates}] \Pr\left[\mathcal{F}^{\mathcal{B}} \text{ wins } \mathsf{c-uf-pres}_{(A)} | \mathcal{B} \text{ simulates}\right] = \\ \Pr[\mathcal{B} \text{ simulates}] \mathsf{Adv}^{\mathsf{c-uf-pres}_{(A)}}_{\mathcal{F},\mathsf{BBS}}(\kappa) \geq \\ \left(1 - \frac{q^2}{p}\right) \mathsf{Adv}^{\mathsf{c-uf-pres}_{(A)}}_{\mathcal{F},\mathsf{BBS}}(\kappa) \quad (10)\end{aligned}$$

Therefore, by the Generalized Forking lemma we have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{DL}}_{\mathcal{B}}(\kappa) \geq \epsilon_A \left(\frac{\epsilon_A}{q_H} - \frac{1}{p}\right) \geq \\ \frac{\left(1 - \frac{q^2}{p}\right)^2}{q_H} \left(\mathsf{Adv}^{\mathsf{c-uf-pres}_{(A)}}_{\mathcal{F},\mathsf{BBS}}(\kappa)\right)^2 - \frac{\left(1 - \frac{q^2}{p}\right)}{p} \left(\mathsf{Adv}^{\mathsf{c-uf-pres}_{(A)}}_{\mathcal{F},\mathsf{BBS}}(\kappa)\right). \quad (11)\end{aligned}$$

G.2 Case B: extract a BBS forgery

Simulation failure probability In this case the reduction $\mathcal{B}$ reduces the hardness of forging a multi-holder presentation to the hardness of forging a BBS signature.

For this case the simulation never fails.

Advantage of the reduction. The advantage of $\mathcal{B}$ in winning the strong unforgeability experiment of the BBS signature is

$$\mathsf{Adv}^{\mathsf{suf-cma}}_{\mathcal{B},\mathsf{BBS}}(\kappa) = \Pr\left[\mathcal{B}^{\mathcal{F}} \text{ wins } \mathsf{suf-cma} \text{ for } \mathsf{BBS}\right]$$

which is the probability that $\mathcal{B}$ interacting with $\mathcal{F}$ wins the strong unforgeability experiment described in [TZ23] for BBS signatures.

As for Case A, we apply the Forking Lemma to the algorithm which is successful if $\mathcal{B}$ correctly simulates (which happens with probability 1) and $\mathcal{F}$ outputs a valid forgery (constructed starting from a credential never issued to $\mathcal{F}$), and we define the probability of success of this algorithm as

$$\epsilon_B = \Pr\left[\mathcal{F}^{\mathcal{B}} \text{ wins } \mathsf{c-uf-pres}_{(B)} \wedge \mathcal{B} \text{ simulates}\right] = \Pr[\mathcal{B} \text{ simulates}] \Pr\left[\mathcal{F}^{\mathcal{B}} \text{ wins } \mathsf{c-uf-pres}_{(B)} | \mathcal{B} \text{ simulates}\right] = \mathsf{Adv}^{\mathsf{c-uf-pres}_{(B)}}_{\mathcal{F},\mathsf{BBS}}(\kappa) \quad (12)$$

By applying the Generalised Forking Lemma [BN06] we obtain

$$\mathsf{Adv}^{\mathsf{suf-cma}}_{\mathcal{B},\mathsf{BBS}}(\kappa) \geq \epsilon_B \left(\frac{\epsilon_B}{q_H} - \frac{1}{p}\right) \geq \frac{\left(\mathsf{Adv}^{\mathsf{c-uf-pres}_{(B)}}_{\mathcal{F},\mathsf{BBS}}(\kappa)\right)^2}{q_H} - \frac{\left(\mathsf{Adv}^{\mathsf{c-uf-pres}_{(B)}}_{\mathcal{F},\mathsf{BBS}}(\kappa)\right)}{p}. \quad (13)$$

H Reducing the Size of Credential Shares

We describe an optimization that can be used to reduce the size of the credential shares which only requires to increase the size of the first broadcast message in the presentation protocol execution.

Optimization 1 (Issuing algorithm $\mathsf{CredIss}_{\mathsf{BBS}}$). *In order to reduce the size of the share of credential, which is polynomial in the number n of participants, the issuer could sign the values D_i and issue to each party P_i the partial credential $\mathsf{cred}_i = (A, e^{(i)}, \sigma_i, D_i, (\{a_j\}_{j\in\mathsf{Pub}}, \{a_j^{(i)}\}_{j\in\mathsf{Prv}}))$, where, being $h = H(X_2, \mathsf{pp}, A, \{a_j\}_{j\in\mathsf{Pub}})$ the hash of the credential data which are shared by all the participants, σ_i is a digital signature $\sigma_i = \mathsf{Sign}(h||D_i, \mathsf{sk})$ ($D_i = \left(\prod_{j\in\mathsf{Prv}} h_j^{a_j^{(i)}}\right) A^{-e^{(i)}}$) created using a secret key sk possessed by the issuer. Then, in the third step of the protocol the parties taking part to the presentation protocol execution broadcast their com_i together with D_i and the signature σ_i of the issuer. In this way they prove that they are using a share of credential which has been certified by the issuer. Then in order to successfully execute the presentation protocol the holder must prove knowledge of the associated integers $e^{(i)}, \{a_j^{(i)}\}_{j\in\mathsf{Prv}}$ s.t. $D_i = \left(\prod_{j\in\mathsf{Prv}} h_j^{a_j^{(i)}}\right) A^{-e^{(i)}}$.*

If we consider a MHAC scheme compatible with a secure anonymous credential scheme, if the issuer only issues full credential we have already described how an holder can distribute the credential issued by the issuer among multiple holders by performing the secret sharing of the credential, giving a share to each holder and then deleting the credential issued. In this case, the holder performing the secret sharing of the BBS *credential generates a key pair* $(\mathsf{sk}_h, \mathsf{pk}_h)$*, signs the values* $\tilde{A}_i$ *using a secret key* sk_h *obtaining* σ_i*, and sends the pair* $(\sigma_i, \mathsf{pk}_h)$ *to each holder* P_i*. Then it deletes the* BBS *credential and the secret key* sk_h*. The protocol participants will broadcast in the first step the signature of* D_i *and the other participants can verify it using* pk_h*.*

I Pedersen Verifiable Secret Sharing

In this section, we describe how it is possible to generate multi-holder anonymous credentials with private attributes using as a building block the verifiable secret sharing by Pedersen [Ped91]. We will consider three cases: the first one where a private attribute is chosen by a dealer who is the issuer; the second case in which the attribute is chosen by a dealer who is a holder who wants to keep it hidden also from the issuer; the third where the private attribute is generated by the holders (and possibly also by the issuer) and remains unknown by both the holders and the issuer.

We will consider that the private attribute is the one in position $m-1$. Using the same notation as in [Ped91], we define the Pedersen commitment to a^* calculated using randomness $s \xleftarrow{\$} \mathbb{Z}_p$ and bases h_{m-1}, h_m as

$$E(a^*, s) = h_{m-1}^{a^*} h_m^s.$$

I.1 Holder as a dealer

The dealer who wants to distribute the private attribute a^* performs the following operations.

1. The dealer computes a Pedersen commitment to a^* sampling $s \xleftarrow{\$} \mathbb{Z}_p$ and computing $D = E(a^*, s)$.
2. Being n the number of holders of a $(t, n)-$multi-holder anonymous credential, the dealer samples at random a polynomial

   $$F(x) = F_0 + F_1 x + \cdots + F_{t-1} x^{t-1} \xleftarrow{\$} \mathbb{Z}_p[x]$$

   of degree at most $t-1$, and such that $F(0) = F_0 = a^*$, and uses this polynomial to generate a Shamir-secret sharing [Sha79] of a^*, setting $a^{*(i)} = F(i), \forall i \in [n]$.
3. The dealer additionally generates another random polynomial of degree at most $t-1$:

   $$G(x) = G_0 + G_1 x + \cdots, + G_{t-1} x^{t-1} \xleftarrow{\$} \mathbb{Z}_p[x]$$

with $G(0) = G_0 = s$, and uses it to compute a secret sharing of the randomness s of the commitment D. It generates the secret shares $s^{(i)} = G(i), \forall i \in [n]$ and sends secretly to P_i the pair $(a^{*(i)}, s^{(i)})$.
4. The dealer also computes the commitments $E_i = E(F_i, G_i)$ to $F_i, \forall i \in \{0\} \cup [k-1]$ and broadcast E_i to all the participants and to the issuer (note that $E_0 = D$).

Share verification. Each holder P_i upon receiving $(a^{*(i)}, s^{(i)}), \{E_i\}_{i \in \{0\} \cup [k-1]}$, checks that

$$E(a^{*(i)}, s^{(i)}) = \prod_{j=0}^{k-1} E_j^{i^j}$$

and if it does not hold, sends an abort message.

Note that if nobody aborts, then everyone can compute the values $E(a^{*(i)}, s^{(i)})$.

Every party creates a straight-line extractable [Pas03,Fis05,Unr15,KS22] NIZKP of knowledge of a representation of $E(a^{*(i)}, s^{(i)})$, then the issuer computes the shares of credential executing Protocol 1.

The unlinkability with private attributes can be instantiated assuming that the challenger of the experiment controls both the issuer of the credential, an the dealer.

I.2 Issuer as a dealer

In this case, the issuer performs the operations performed by the holder in the previous case (Section I.1), and in this case, it does not have to sign in a blind way any attribute since it also knows the private attributes.

I.3 Issuance without dealer

In this section we instantiate the protocol for the generation of an anonymous shared secret proposed by Pedersen in [Ped91, Section 5.2] in the context of our multi-holder issuing protocol, showing how the parties can generate a shared attribute a^* (and the masking attribute s corresponding to the $m-$th position) which is not known by anyone.

Each holder P_i (and possibly the issuer) executes the following operations:

- sample $a_i^* \xleftarrow{\$} \mathbb{Z}_p$ uniformly at random;
- distribute a_i^* as if it was the dealer using the Pedersen VSS described in Section I.1 and signs each share $(a_i^{*(j)}, s_i^{(j)}) \forall j \in [n] \setminus \{i\}$ that sends to the parties $\{\mathsf{P}_j\}$ and the broadcasted values $(E_{i,0}, \cdots, E_{i,t-1})$;
- P_i verifies the shares received by the other parties, and if one does not verify, P_i broadcast the share with the signature of the associated dealer and aborts.
- compute the share $(a^{*(i)}, s^{(i)})$ of $a^* = \sum_{j \in [n]} a_j^*$ setting $a^{*(i)} = \sum_{j \in [n]} a_j^{*(i)}$ and $s^{(i)} = \sum_{j \in [n]} s_j^{(i)}$, then by computing

$$(E_0, \cdots, E_{t-1}) = (\prod_{j \in [n]} E_{j,0}, \cdots, \prod_{j \in [n]} E_{j,t-1})$$

Every party creates a straight line extractable [Pas03,Fis05,Unr15,KS22] NIZKP of knowledge of a representation of $E(a^{*(i)}, s^{(i)})$, then the issuer computes the shares of credential executing Protocol 1.