# Multi-Holder Anonymous Credentials from BBS Signatures

Andrea Flamini<sup>1</sup>, Eysa Lee<sup>2</sup>, and Anna Lysyanskaya<sup>3</sup>

<sup>1</sup> Mathematics Department, University of Trento, Povo, Trento, Italy andrea.flamini@unitn.it

<sup>2</sup> Data Science Institute, Brown University, Providence, RI, USA eysa\_lee@brown.edu

<sup>3</sup> 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

M	ulti-Holder Anonymous Credentials from BBS Signatures				
Andrea Flamini, Eysa Lee, and Anna Lysyanskaya					
1					
	1.1 Our Contribution	2			
	1.2 Our Techniques	3			
	1.3 Outline	5			
2	Related Works	6			
3	Preliminaries	7			
	3.1 Sigma protocols	8			
	3.2 BBS signatures	9			
	Zero-Knowledge Proofs of Knowledge for BBS Signatures	9			
4	Multi-Holder Anonymous Credentials	10			
<b>5</b>					
	5.1 Correctness	11			
	5.2 Unlinkability	12			
	Unlinkability of presentations	12			
	Unlinkability of private attributes.	13			
	5.3 Presentation with identifiable abort	14			
	5.4 Concurrent unforgeability of presentations	15			
6	BBS Multi-Holder Anonymous Credentials	18			
	6.1 Credential issuing	18			
	6.2 Multi-holder presentation	20			
	6.3 Verification	23			
	6.4 Extensions	23			
7	Security Analysis	24			
	7.1 Correctness of BBS MHAC	24			
	7.2 Unlinkability of presentations of BBS MHAC	25			
	7.3 Unlinkability of private attributes	25			
	7.4 Presentation with identifiable abort of BBS MHAC	26			
	7.5 Unforgeability of presentations of BBS MHAC	26			
	Acknowledgments	27			
Α	Sigma Protocol for Linear Relation	30			
В	Analysis of BBS Presentation Protocol in [TZ23].	30			
С	BBS MHAC Presentation Protocol Overview	35			
D	Private Attribute Unlinkability of BBS MHAC Scheme	37			
Е	•				
F	Unforgeability of the BBS MHAC Scheme	39			
	F.1 Unforgeability proof	41			
	Case A: reduction to DL	42			
	Case B: reduction to $suf - cma$ for BBS	48			
G	Advantage of the Presentation Unforgeability Reduction	51			

		Case A: extract the target credential Case B: extract a BBS forgery	
Н		icing the Size of Credential Shares	
Ι	Pede	rsen Verifiable Secret Sharing	55
	I.1	Holder as a dealer	56
	I.2	Issuer as a dealer	56
	I.3	Issuance without dealer	57

# 1 Introduction

According to W3C Verifiable Credential Data  $Model^4$ , "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<sup>5</sup>, 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<sup>+</sup>24] recommends the creation of the 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 Camenish and Lysyanskaya [CL04] suggested could be adapted to anonymous credentials. The resulting scheme (and a variant called BBS+) was subsequently analyzed, improved, and adapted, in a provably secure fashion, for use in direct anonymous attestation (DAA) and anonymous credential schemes [ASM06,BL10,CDL16].

<sup>&</sup>lt;sup>4</sup> https://www.w3.org/TR/vc-data-model-2.0/

<sup>&</sup>lt;sup>5</sup> https://digital-strategy.ec.europa.eu/en/policies/eidas-regulation

The state-of-the-art proof of security 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]<sup>6</sup> and is also the object of a standardization effort from W3C [LKWL22]<sup>7</sup>.

*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. 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 it 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 tdevices will be unable to demonstrate possession of the credential or even learn anything about the secret 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.

Let us go over unforgeability for MHAC in more depth. Suppose an adversary controls fewer than t holders of a credential with attributes **a** issued by an

<sup>&</sup>lt;sup>6</sup> A comparison between [CL03,PS16,CDL16,TZ23] is performed in [FSS<sup>+</sup>24]

<sup>&</sup>lt;sup>7</sup> 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 been recently included in an update of the paper of [TZ23, Appendix B].

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*. The adversary can also query to see presentations of credentials with any attribute sets it chooses. 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 not learn anything it doesn't 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 an 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 **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  $\mathbf{e}(C(\mathbf{a}), (\mathsf{pk})g_2^e) = \mathbf{e}(A, g_2)$ , 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 **a** and the BBS signature on this vector can be recovered as follows: set  $r = \alpha$ , let  $m_i = \beta_i/\alpha$ , and let  $e = -\gamma$ .

As a result, a zero-knowledge proof of knowledge of the message vector **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 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 signature.

**Credential secret sharing.** How do we secret-share credential attributes and a BBS anonymous credential in such a way that the protocol used to create a presentation is efficient? Our approach is to secret-share the value e of the BBS signature, as well as those of the attributes  $\{a_j\}_{j\in\mathsf{Prv}}$  in **a** that are especially sensitive and that we don't want to store in the clear on any device. The remaining attributes in **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. 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 **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  $D = (\prod_{j \in \mathsf{Prv}} h_j^{a_j}) A^{-e}$  is (implicitly) provided to all the holders. To be more precise, we give to each holder  $\{D_i\}_{i \in [n]}$ , with  $D_i = (\prod_{j \in \mathsf{Prv}} h_j^{a_j^{(i)}}) A^{-e^{(i)}}$ , 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  $\{a_j\}_{j \in \mathsf{Prv}}$ .

One of the most tricky aspects of the proof of security for our scheme is showing that it is safe to reveal D. We show that an adversary that breaks unforgeability of our scheme will either lead to a reduction that forges a BBS signature, or to another reduction that is set up such that it knows the BBS signing key, but ends up breaking the discrete logarithm problem. This second reduction receives as input the DL challenge (g, h), and generates the values  $A, \tilde{A} = A^e$  such that  $\log_A \tilde{A} = \log_g h$ ; thus if the adversary succeeds in forging a proof of knowledge of a 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  $\mathsf{P}_j$ , to sample a random  $r \stackrel{\$}{\leftarrow} \mathbb{Z}_p$  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.

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

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  $\mathsf{P}_j$ ;  
-  $\tilde{B}_i = D_i^{r\lambda_{S,i}(0)}, \forall i \in S \setminus \{j\}$ , 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  $\mathsf{P}_j$  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 [CKM23] 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], Keller, Mikkelsen and Rupp describe a framework for distributing the prover side of sigma protocols over multiple parties. Their framework requires (1) the presence of a (possibly malicious) combiner which aggregates the information generated by the provers, and (2) the interaction between the combiner and a (possibly malicious) verifier during the execution of the proof of knowledge protocol. Also, they assume that the adversary can corrupt less than t-out-of-n parties (as we do), and can have some additional control over the combiner and the verifier. One of the examples of application of their framework that the authors include is a multi-party version of the sigma protocol for proving knowledge of a CL anonymous credential [CL04].

Instead in our work we will consider protocols that are non-interactive with respect to the verifier, meaning that the provers interact for the creation of the credential presentation and send the protocol output to the verifier.

The distribution of the prover side of (threshold) sigma protocols made noninteractive with respect to the verifier has been studied in [BF24], where the authors describe a framework for the creation of threshold signature schemes starting from threshold sigma protocols similar to the ones described in [KMR12].

Our work follows the setting adopted in [BF24] and more generally by threshold digital signatures [CKM23,DKL<sup>+</sup>23], and we design a protocol which does not require the interaction between the provers and the verifier. We also do not require the existence of the combiner since we assume a broadcast channel between the provers. Therefore in our security analysis we do not need to consider the case in which the verifier is malicious and we only focus on:

- 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 credential 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  $x \stackrel{\$}{\leftarrow} S$  denote sampling an element x from a set S uniformly randomly. Let  $x \stackrel{\$}{\leftarrow} A(i_1, \ldots, i_n)$  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 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.

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  $\psi$ .

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  $\mathsf{P}_i$  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,  $\{p(i)\}_{i\in[n]} \stackrel{\$}{\leftarrow} \mathsf{Share}(t, n, v)$ , 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  $pp \leftarrow (\mathbb{G}, p, g)$  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  $\mathcal{A}$ 

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

where  $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p, (g, g^x) \in \mathbb{G}^2$  and  $\kappa$  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  $\mathcal{A}$ 

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

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  $\kappa$  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  $\pi$  for the relation  $\mathcal{R}$  is an interactive protocol between a prover  $\mathsf{P}$  and a verifier  $\mathsf{V}$  with 3-move form, i.e.  $\mathsf{P}$  sends a commitment cmt to  $\mathsf{V}$ , who replies to  $\mathsf{P}$  with a random challenge ch, and finally  $\mathsf{P}$  computes a response rsp that is sent back to  $\mathsf{V}$ .  $\pi$  also satisfies the following properties:

- $-\pi$  has completeness, which means that if P and V execute  $\pi$  with common input y and private input w to P, with  $(w, y) \in \mathcal{R}$ , V always accepts;
- $\pi$  has special soundness, which means that, from any y, and from a pair of valid conversations for input y, (cmt, ch, rsp), (cmt, ch', rsp') with ch  $\neq$  ch', one can easily compute w s.t.  $(w, y) \in \mathcal{R}$ ;
- $-\pi$  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 (cmt, ch, 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 Camenish 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  $h_1, \ldots, h_m \stackrel{\$}{\leftarrow} \mathbb{G}_1$  and set the set of public parameters  $\mathsf{pp} \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2, h_1, \ldots, h_m)$ .
- KGen<sub>BBS</sub>(pp). Sample a random  $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ . Compute  $X_2 = g_2^x$ , and set  $\mathsf{sk} \leftarrow x$ , and  $\mathsf{pk} \leftarrow X_2$ .
- Sign<sub>BBS</sub>(pp, sk,  $(a_1, \ldots, a_m)$ ). Compute  $C(\mathbf{a}) = g_1 \prod_{i=1}^m h_i^{a_i}$ . Randomly generate  $e^{\frac{S}{2}} \mathbb{Z}$  and compute  $A = C(\mathbf{a})^{\frac{1}{e+x}}$ . Output the pair  $(A, e) \in \mathbb{C}_{r_i} \times \mathbb{Z}$ .
- erate  $e \stackrel{\$}{\leftarrow} \mathbb{Z}_p$  and compute  $A = C(\mathbf{a})^{\frac{1}{e+x}}$ . Output the pair  $(A, e) \in \mathbb{G}_1 \times \mathbb{Z}_p$ . - Verify<sub>BBS</sub>(pp, 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 Protocol 3 and Figure 2 in Appendix B.

This protocol is a proof of knowledge of a BBS signature<sup>8</sup> 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  $\text{Rev} \subseteq [m]$ , and to the hidden attribute with the symbol  $\text{Hid} = [m] \setminus \text{Rev}$ .

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.,  $\mathbf{e}(\overline{A}, X_2) = \mathbf{e}(\overline{B}, g_2)$ ).

<sup>&</sup>lt;sup>8</sup> This statement is true if we adopt this protocol in the context of an anonymous credential scheme, where it is assumed that the signer (the issuer) generates and sends to the holders only valid signatures, as we elaborate in Appendix B.

Non-interactive and fresh proofs of knowledge. The sigma protocol in Protocol 3 and Figure 2 can be 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  $\operatorname{cred}_i$  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:

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

This algorithm generates public parameters pp (e.g. the number of attributes m) and the issuer key pair (pk, 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 algorithm is a protocol executed by a set  $\{P_i\}_{i\in S}$  of t holders who jointly create a presentation pres for nonce and public attributes  $\{a_i\}_{i\in Rev}$ . - Multi-holder presentation verification algorithm:

muin-noiaer 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;
- the presentation pres produced by 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. To convert a multi-holder version into the single holder, the issuer can simply send t shares to a single party and the party can generate the presentation on its own. 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).

*Remark 1.* In the definition above, 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 secret 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 secret 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 10), and concurrent unforgeability of presentations (Definition 11).

## 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,  $\{a_i\}_{i \in [m]}$ , Rev  $\subseteq [m] \setminus \text{Prv}, S \subseteq [n], |S| = t, t \leq n, it holds that <math>1 \leftarrow \text{VfPres}(pp, pk, \text{nonce}, \{a_i\}_{i \in \text{Rev}}, \text{pres})$ where

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

## 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 secret attributes (i.e., attributes not known to all holders), an adversary cannot learn any information about these secret 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 ( $Exp_{\mathcal{A}}^{unlink}(\kappa)$ — MHAC Presentation Unlinkability).

- The adversary A generates a set of public parameters pp, an issuer public key pk, and a multi-holder credential {cred<sub>i</sub>}<sub>i∈[n]</sub> on attributes {a<sub>i</sub>}<sub>i∈[m]</sub> of its choosing issued under pk. The adversary sends this information to the challenger C together with the information related to the presentation that C must produce, namely nonce, {a<sub>i</sub>}<sub>i∈Rev</sub> ⊆ {a<sub>i</sub>}<sub>i∈[m]</sub>.
- 2. C runs pres  $\leftarrow$  CredPres(pp, pk, t, {(P<sub>i</sub>, cred<sub>i</sub>)<sub>i \in S</sub>, { $a_i$ }<sub>i \in Rev</sub>, nonce}) with a set  $S \subseteq [n], |S| = t$  and records the transcript of the protocol execution as T. C then checks that VfPres(pp, pk, nonce, { $a_i$ }<sub>i \in Rev</sub>, pres)  $\rightarrow 1$ . If the presentation does not verify, C aborts and the experiment outputs 0. Otherwise, C samples uniformly at random a bit b. If b = 1, C overwrites (pres, T) with the output from a simulated presentation as (pres, T)  $\leftarrow$  SimCredPres(pp, pk,  $t, \tau$ , { $a_i$ }<sub>i \in Rev</sub>, nonce). Otherwise, C keeps (pres, T) as is.
- 3. C sends (pres, T) to the adversary 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

SimCredPres(pp, pk,  $t, \{a_i\}_{i \in \text{Rev}}$ , nonce) such that an adversary  $\mathcal{A}$  can win  $\text{Exp}_{\mathcal{A}}^{\text{unlink}}(\kappa)$  with at most negligible advantage. That is,

$$\left| \Pr \left[ \mathsf{Exp}_{\mathcal{A}}^{\mathsf{unlink}}(\kappa) = 1 \right] - \frac{1}{2} \right| \le \nu(\kappa), \text{ where } \nu(\kappa) \text{ is negligible in } \kappa.$$

Remark 2. Note that if a MHAC scheme is compatible with a secure anonymous credential scheme, then the unlinkability property is derived from the unlinkability of the underlying anonymous credential scheme since the challenger can execute the simulator of the NIZKP as SimCredPres(pp, pk, t,  $\{a_i\}_{i \in \text{Rev}}$ , nonce) to generate a simulated presentation.

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 secret attributes are determined when the credential is issued and are always a subset of the attributes that are hidden from the verifier.

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

- 1. The challenger C runs  $(pp, (pk, sk)) \xleftarrow{\$} \mathsf{IssSetup}(\kappa)$  and sends (pp, (pk, sk)) to the adversary  $\mathcal{A}$ .
- 2. A chooses and sends to C two challenge secret attributes  $a_0^*, a_1^*$ .
- 3. C flips a coin  $b \stackrel{\$}{\leftarrow} \{0,1\}$  and C runs Credlss with A honestly with secret attribute  $a_b^*$  and public attributes of  $\mathcal{A}$ 's choosing<sup>9</sup>. C plays the role of the issuer and the honest parties, and  $\mathcal{A}$  and C receive their respective shares of the credential.
- 4. The adversary may choose to run 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 secret attribute), with C playing the role of the honest parties.
- 5. At the end, A sends C its guess b'. If b = b', the experiment outputs 1, otherwise the experiment outputs 0.

**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,

 $\left| \Pr \Big[ \mathsf{Exp}_{\mathcal{A}}^{\mathsf{unlink-attr}}(\kappa) = 1 \Big] - \frac{1}{2} \right| \le \nu(\kappa), \text{ where } \nu(\kappa) \text{ is negligible in } \kappa.$ 

*Remark* 3. 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

<sup>&</sup>lt;sup>9</sup> Note that the challenger C, which in this experiment acts as an issuer, knows the value of the secret attribute. This is not always true in general, in fact a secret attribute might be unknown both to the holders and to the issuer.

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 4. One might ask what the motivation is behind having secret attributes not known to holders. The secret attributes might be attributes that are sometimes revealed, but only in extremely rare circumstances. In these circumstances, the holders reconstruct the secret attribute, reveal it, prove its correctness, and then erase it. For secret attributes that are never revealed, we can consider a multi-authority scenario in which issuers use a secret attributes 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 secret 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 behaviour 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.

**Definition 10 (Presentation with identifiable abort).** Let CredPres be the multi-holder credential presentation algorithm and let  $\{P_i\}_{i\in S}$  be a subset of t parties executing the credential presentation protocol. Let W be an algorithm which can be executed by any party  $P_i$ ,  $i \in S$ . W is a "wrapper" interactive algorithm that relays messages between CredPres algorithm run by  $P_i$  and the other participants in S, 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.

Each time that  $P_i$  receives a message m from another protocol participant, m is forwarded to W's input tape; W's processing of m results in either forwarding m to CredPres (more precisely, W clears its input tape, updates its state tape, and outputs (message, m) on its output tape, resulting in the message going through), or 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).

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.

We say that CredPres satisfies identifiable abort if there exists an efficient wrapper algorithm W such that the following properties hold for the composed algorithm W(CredPres):

- Correctness: Whenever W did not cause CredPres to abort, the output pres of CredPres verifies, i.e. VfPres(pp, pk, nonce,  $\{a_i\}_{i \in \text{Rev}}$ , pres) = 1.
- Identifiability: Whenever the output of W is (abort, j) for some  $j \in S$ ,  $\mathsf{P}_j$  did not follow the protocol instructions and is therefore corrupt.

*Remark 5.* 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.

# 5.4 Concurrent unforgeability of presentations

We describe an experiment defining the unforgeability of a multi-holder anonymous credential presentation algorithm CredPres. The experiment resembles the security experiment for threshold signature schemes. We can think of the 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 2 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  $Exp_{\mathcal{A}}^{c-uf-pres}$  is given below and summarized in Figure 5.

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

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

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

- $\mathcal{F}$  can query an issuing oracle  $\mathcal{O}_{iss}$  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)  $\{\operatorname{cred}_i\}_{i \in [cor]}$  for  $\{a_i\}_{i \in [m]}$ .<sup>10</sup>
  - Issuance of full credentials: on input the set of attributes {a<sub>i</sub>}<sub>i∈[m]</sub> chosen by F, O<sub>iss</sub> provides F with a credential cred = {cred<sub>i</sub>}<sub>i∈[n]</sub> <sup>\$</sup> CredIss(pp, sk, n, t, {a<sub>i</sub>}<sub>i∈[m]</sub>, Prv) on these attributes.

 $\mathcal{O}_{iss}$  stores a record (cred) in a credential table CT.

Issuance of the target credential: F gives as input to O<sub>iss</sub> the tuple ({a<sub>i</sub>}<sub>i∈[m]</sub>, t, n, cor) where {a<sub>i</sub>}<sub>i∈[m]</sub> are attributes chosen by F to include in the credential, (t, n) are the parameters of the secret sharing of the credential, and cor ⊂ [n], |cor| < t, are the parties F wants to corrupt.</li>
O<sub>iss</sub> computes Credlss(pp, sk, t, n, {a<sub>i</sub>}<sub>i∈[m]</sub>, Prv) → {cred<sub>i</sub>}<sub>i∈[n]</sub> and gives

to  $\mathcal{F}$  only the shares corresponding to the parties in cor.

 $\mathcal{O}_{\mathsf{iss}}$  stores the value targetCred  $\leftarrow (\{\mathsf{cred}_i\}_{i \in [n]}, n, t, \mathsf{cor}).$ 

- $\mathcal{F}$  can query a presentation oracle  $\mathcal{O}_{pres}$  for a polynomial number  $q_P$  of presentations specifying the nonce nonce to use and the attributes  $\{a_i\}_{i\in\mathsf{Rev}}$  to reveal.  $\mathcal{F}$  can query  $\mathcal{O}_{pres}$  for two kind of presentations: the presentation of the target credential, or of a credential it does not control at all.<sup>11</sup> We allow the adversary to open concurrently many session 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  $\mathcal{O}_{pres}$ . Since ssid in each message exchanged by the protocol participants, we will omit it, and we will explicitly write only in the oracle queries made by the adversary to the presentation oracle.
  - Presentation of the target credential targetCred: F gives in input to O<sub>pres</sub> the tuple (ssid, nonce, {a<sub>i</sub>}<sub>i∈Rev</sub>, hon) which specifies the nonce for the presentation, the set of attributes to reveal, the set of parties hon ⊆ [n] \ cor

<sup>&</sup>lt;sup>10</sup> 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.

<sup>&</sup>lt;sup>11</sup> Note that  $\mathcal{F}$  can generate presentations for the full credentials on its own, without the help of any oracle.

s.t. |cor| + |hon| = t and the session identifier ssid which identifies the specific presentation protocol execution session.

 $\mathcal{O}_{pres}$ , acting on behalf of the parties in hon, interacts with  $\mathcal{F}$ , who controls the parties in cor, in the execution of  $CredPres(\{P_i, cred_i\}_{i \in cor \cup hon}, \{a_i\}_{i \in Rev}, nonce, pp, pk)$  with session identifier ssid.

If  $\mathcal{O}_{pres}$  sends its last protocol message associated to the protocol execution identified by ssid, it stores in the presentation table PT the record (nonce,  $\{a_i\}_{i \in \text{Rev}}, \bot$ ).

The last entry refers to the output of the presentation protocol, but  $\mathcal{O}_{\text{pres}}$  is not guaranteed to learn the output since in the protocol execution it always sends its messages first<sup>12</sup>.

Presentation of a credential never issued to F: F only gives in input to O<sub>pres</sub> the tuple (nonce, {a<sub>i</sub>}<sub>i∈Rev</sub>, t, n) (this input does not specify the set hon), and receives a presentation pres generated autonomously by O<sub>pres</sub> who stores in PT the record (nonce, {a<sub>i</sub>}<sub>i∈Rev</sub>, pres).

The presentation pres is generated by firstly generating a (t, n)-MHAC cred for the attributes  $\mathbf{a} = \{a_i\}_{i \in \mathsf{Rev}} \cup \{r_i\}_{i \in [m] \setminus \mathsf{Rev}}$ , where the attributes  $r_i$  with  $i \in [m] \setminus \mathsf{Rev}$  are sampled uniformly at random in  $\mathbb{Z}_p$ , executing cred  $\stackrel{\$}{\leftarrow}$  Credlss(sk, pp,  $t, n, \mathbf{a}, \mathsf{Prv})$  and then by creating a presentation of the credential executing pres  $\stackrel{\$}{\leftarrow}$  CredPres( $\{(\mathsf{P}_i, \mathsf{cred}_i)\}_{i \in [t]}, \{a_i\}_i \in \mathsf{Rev}, \mathsf{nonce}, \mathsf{pp}, \mathsf{pk}\}$ .

Forgery phase. At the end of the training,  $\mathcal{F}$  must produce a forgery (nonce<sup>\*</sup>,  $\{a_i^*\}_{i \in \text{Rev}^*}$ , pres<sup>\*</sup>) given by a presentation pres<sup>\*</sup> for (nonce<sup>\*</sup>,  $\{a_i^*\}_{i \in \text{Rev}^*}$ ) of its choice.

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

- For every record (nonce,  $\{a_i\}_{i \in \mathsf{Rev}}$ , pres) in the presentation table PT the following holds:
  - if pres ≠ ⊥: (nonce<sup>\*</sup>, {a<sub>i</sub><sup>\*</sup>}<sub>i∈Rev<sup>\*</sup></sub>, pres<sup>\*</sup>) ≠ (nonce, {a<sub>i</sub>}<sub>i∈Rev</sub>, pres)
     (i.e. the record is associated to the presentation of a credential never issued to F).
  - if pres = ⊥ (i.e. the record is associated to a presentation of the target credential) we require that (nonce<sup>\*</sup>, {a<sub>i</sub><sup>\*</sup>}<sub>i∈Rev<sup>\*</sup></sub>) ≠ (nonce, {a<sub>i</sub>}<sub>i∈Rev</sub>).
- For every record cred in CT, being  $\{a_i\}_{i \in [m]}$  the attributes associated to cred,  $\{a_i\}_{i \in \text{Rev}^*} \neq \{a_i^*\}_{i \in \text{Rev}^*}$ .

This guarantees that the forgery is not derived from a full credential that has been issued by  $\mathcal{O}_{iss}$ .

**Definition 11 (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  $\kappa$ .

<sup>&</sup>lt;sup>12</sup> Note that storing cor is not useful to the scopes of checking the validity of the forgery, since a forgery do not specify the actors involved in the creation of it.

**Observation 1.** 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 2.** Note that we could allow the adversary of the unforgeability game to be issued 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  $\kappa$ . This would impact the dimension of the parameters when it comes the time to instantiate the scheme.

Remark 6. The only kind of queries allowing the adversary to open many concurrent sessions are the queries for the presentation of the target credential. The reason is that the presentation queries for credentials the adversary was never issued does not require the interaction of the adversary with  $\mathcal{O}_{\text{pres}}$ , moreover, for what concerns the full credentials  $\mathcal{F}$  has been issued, the adversary could simulate the opening of concurrent presentation sessions since it knows t shares of the credential, therefore it would also know all the secret shares of the parties it does not control.

# 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).

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 is 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 the one party always participate 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.  $\operatorname{Pgen}_{\operatorname{BBS}}(\kappa) \to \operatorname{pp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2, h_1, \dots, h_m)$
- 2. KGen<sub>BBS</sub>(pp)  $\rightarrow$  (sk, pk) = (x, g\_2^x)
- *3.* Output (pp, (pk, sk))

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

In the case where some attributes may not be known to all holders, each party's credential will have a share of the private attribute rather than full attribute itself. Our starting point here is multi-base Pedersen verifiable secret sharing (VSS) [Ped91]<sup>13</sup>, 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  $a_j$ , each party  $\mathsf{P}_i$ '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  $\mathsf{P}_i$  also has a Shamir share of it,  $a_m^{(i)}$ . To simplify our notation we will include m in the set of private attributes  $\mathsf{Prv}$ . Finally, for each  $\mathsf{P}_i$ , 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<sup>14</sup>. We assume this was setup prior to the protocol's execution, and that each holder has also published a straight-line extractable [Fis05,KS22,LR22] proof of knowledge  $\pi_i$  of these secret shares.

Overview of issuing with private attributes. From the multi-base Pedersen commitments  $\{C_i\}$  and the public attributes, the issuer (after verifying the proofs  $\pi_i$ ) computes  $C(\mathbf{a})$ , and then computes the rest of the signature: it picks a random e, computes  $A = C(\mathbf{a})^{1/(x+e)}$ , secret-shares e into  $\{e_i\}$ , and distributes the resulting shares.

Protocol 1. Credlss<sub>BBS</sub> — Multi-holder issuing protocol

**Credential Issuance without Private Attributes.** Upon receiving attributes  $\{a_i\}_{i \in [m]}$ , the issuer creates a credential as follows.

1. Compute a BBS signature as  $(A, e) \xleftarrow{\$} \mathsf{Sign}_{\mathsf{BBS}}(\mathsf{sk}, \{a_i\}_{i \in [m]})$ 

<sup>&</sup>lt;sup>13</sup> 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 secret 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, Section5.2], and in this case not even the issuer will know this value.

<sup>&</sup>lt;sup>14</sup> Note that  $C = \prod_{j \in \mathsf{Prv} \setminus \{m\}} h_j^{a_j} h_m^{a_m}$  for a random  $a_m$  is a Pedersen commitment to  $\{a_j\}_{j \in \mathsf{Prv} \setminus \{m\}}$ , and therefore  $C_i$  are a shares of C.

2.  $\{e^{(i)}\}_{i \in [n]} \stackrel{\$}{\leftarrow} \text{Share}(t, n, e)$ 

3. For  $i \in [n]$ , set  $\operatorname{cred}_i = (A, e^{(i)}, \{D_j\}_{j \in [n] \setminus \{i\}}, (\{a_j\}_{j \in [m]}, \bot))$  with  $D_j = A^{-e^{(j)}}$  and output  $\operatorname{cred}_i$  to party  $\mathsf{P}_i$ . Credential Issuance with Private Attributes. Let  $\mathsf{Prv}$  denote the set

- Credential Issuance with Private Attributes. Let  $\Pr v$  denote the set of private attributes and  $\operatorname{Pub}$  the set of attributes known by each holder, with  $[m] = \Pr v \cup \operatorname{Pub}$ . We assume that private attributes have already been shared between holders using Pedersen VSS [Ped91]. As a result, for each private attribute  $a_j$ , each party  $\operatorname{P}_i$ 's share of the credential contains a Shamir secret share  $a_j^{(i)}$ ; additionally, the  $m^{th}$  attribute is always a private attribute that is meant to serve as the randomness for Pedersen VSS, so  $\operatorname{P}_i$  also has a Shamir share of it,  $a_m^{(i)}$ , and in particular if  $\operatorname{Prv} \neq \emptyset$ , then  $m \in \operatorname{Prv}^{15}$ . Finally, for each  $\operatorname{P}_i$ , a multi-base Pedersen commitment  $C_i = \prod_{j \in \operatorname{Prv}} h_j^{a_j^{(i)}}$  and a straight-line extractable proof of knowledge  $\pi_i$  of these shares is known to these attribute shares is known. To create a credential with private attributes, the issuer performs the following:
  - 1. For each  $\mathsf{P}_i$ , verify proof  $\pi_i$  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  $C(\mathbf{a}) = g_1 C \prod_{j \notin \mathsf{Prv}} h_j^{a_j}$ . Pick a random *e* and compute  $A = C(\mathbf{a})^{1/(x+e)}$ .
  - 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  $P_i$ .

## 6.2 Multi-holder presentation

An overview of the presentation protocol is depicted in Figure 4. For simplicity, we describe Protocol 2 in terms of attributes either known to all parties (denoted by the set Pub) or shared among all parties (denoted by the set Prv). These sets may be different than the set of attributes that is revealed to the verifier in the credential presentation, which is denoted as Rev. 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.

Protocol 2. CredPres<sub>BBS</sub> — Multi-holder presentation protocol

This protocol is run with parties  $P_1, \ldots, P_n$ , with each party  $P_i$  holding a share of a credential

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

Let  $S \subseteq [n]$  denote a set of party indices, where |S| = t and  $j \in S$  refers to a designated "primary" party  $\mathsf{P}_j$ . Upon receiving the nonce nonce from a verifier for presenting a credential with set of revealed attributes  $\mathbf{a}' = \{a_j\}_{j \in \mathsf{Rev}}$ , parties  $\mathsf{P}_i$  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  $\mathsf{P}_j$  first samples an element  $r \stackrel{\$}{\leftarrow} \mathbb{Z}_p$  broadcasts r to every other party in S.
- 2. Every party  $\mathsf{P}_i$  for  $i \in S$  computes:

$$\begin{split} \overline{A} &= A^r, \quad D = \prod_{k \in S} D_k^{\lambda_{S,k}(0)}, \quad C(\mathbf{a}') = g_1 \prod_{j \in \mathsf{Rev}} h_j^{a_j}, \\ \widetilde{B}_j &= \left( C(\mathbf{a}') \cdot \left(\prod_{k \in \mathsf{Hid} \backslash \mathsf{Prv}} h_k^{a_k}\right) \cdot D_j^{\lambda_{j,S}(0)} \right)^r, \quad \widetilde{B}_i = \left(D_i^{\lambda_{i,S}(0)}\right)^r, \\ \overline{B} &= \prod_{i \in S} \widetilde{B}_i = \left( C(\mathbf{a}') \cdot \left(\prod_{k \in \mathsf{Hid} \backslash \mathsf{Prv}} h_k^{a_k}\right) \cdot D \right)^r. \end{split}$$

where  $\lambda_{S,i}(0)$  denotes the Lagrange coefficient for interpolating party  $\mathsf{P}_i$ 's share with the parties indexed by S.

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:
  - $\mathsf{P}_j$  samples  $\alpha^{(j)}, \{\beta_i^{(j)}\}_{i \in \mathsf{Hid}}, \gamma^{(j)} \stackrel{\$}{\leftarrow} \mathbb{Z}_p$  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  $\mathsf{P}_k$  for  $k \in S \setminus \{j\}$  instead samples  $\{\beta_i^{(k)}\}_{i \in \mathsf{Prv}}, \gamma^{(k)} \stackrel{\$}{\leftarrow} \mathbb{Z}_p$  and computes

$$U_k = \prod_{i \in \mathsf{Prv}} h_i^{\beta_i^{(k)}} \cdot \overline{A}^{\gamma^{(k)}}$$

All the participants  $\mathsf{P}_i$ , 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  $\mathsf{com}_i$  to the other parties.

- 4. Upon receiving  $\operatorname{com}_k$  from every other party  $k \in S \setminus \{i\}$ , each  $\mathsf{P}_i$  opens its commitment by broadcasting  $U_i$  to every other party.
- 5. For each  $U_k$  party  $\mathsf{P}_i$  receives from each  $\mathsf{P}_k$  for  $k \in S \setminus \{i\}$ , if  $U_k$  is not a valid opening for  $\mathsf{com}_k$ , then  $\mathsf{P}_i$  outputs (abort, k) and aborts.
- 6. For each  $k \in S$ ,  $\mathsf{P}_k$  computes:

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

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

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

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

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

$$U_j \cdot \tilde{B}^{\mathsf{ch}}_j \stackrel{?}{=} C(\mathbf{a}')^{z^{(j)}_r} \cdot \prod_{i \in \mathsf{Hid}} h^{z^{(j)}_i}_i \cdot \overline{A}^{z^{(j)}_e}$$

If the equality does not hold, then  $\mathsf{P}_i$  outputs (abort, j) and aborts. Otherwise, upon receiving  $z_e^{(k)}, \{z_i^{(k)}\}_{i\in\mathsf{Prv}}$  from party  $\mathsf{P}_k$  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  $\mathsf{P}_i$  outputs (abort, k) and aborts. 8. For each  $k \in S$ , party  $\mathsf{P}_k$  computes

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

where j corresponds to the index of the primary.  $\mathsf{P}_k$  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 (nonce, 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 different subset of Hid  $\cap$  Pub than the one agreed with the other parties taking part to the presentation protocol.

# 6.3 Verification

#### Algorithm 2 (Multi-holder presentation verification algorithm).

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

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

$$\begin{split} U &\leftarrow \overline{B}^{-\mathsf{ch}} C(\mathbf{a}')^{z_r} \prod_{i \in \mathsf{Hid}} h_i^{z_i} \overline{A}^{z_e}, \\ \mathsf{ch} \stackrel{?}{=} \mathcal{H}_{\mathsf{sig}}\left(\mathsf{ssid}, \mathsf{nonce}, U, \overline{A}, \overline{B}, \{a_i\}_{i \in \mathsf{Rev}}\right), \quad \mathbf{e}(\overline{A}, X_2) \stackrel{?}{=} \mathbf{e}(\overline{B}, g_2) \end{split}$$

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

Remark 7.  $\Pi_{MHAC-BBS} = (IssSetup_{BBS}, CredIss_{BBS}, CredPres_{BBS}, VfPres_{BBS})$  is compatible with the BBS anonymous credential scheme presented in [TZ23], in fact the credential issuing algorithm CredIss\_{BBS} is obtained by generating a BBS anonymous credential and by computing a secret sharing of it. Moreover, the presentation algorithm CredPres\_{BBS} returns a presentation which has the same structure and, assuming the participants to be honest, it has the same distribution of the presentations of BBS anonymous credentials.

#### 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 whose size is linear in the number n of participants. This is because every party knows the values  $D_i$  of the other group members. The idea is that the issuer gives each party  $\mathsf{P}_i$  only its own values  $D_i$  together with a signature  $\sigma_i = \mathsf{Sign}(\mathsf{sk}, D_i, \cdot)$  (where  $\mathsf{sk}$  is the secret key of the issuer) of  $D_i$  (and some values binding this value to the multi-holder anonymous credential). In the first step of the presentation protocol the participants broadcast their values  $\mathsf{com}_i$  together with the values  $(D_i, \sigma_i)$  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  $\Pi_{MHAC-BBS} = (IssSetup_{BBS}, CredIss_{BBS}, CredPres_{BBS}, VfPres_{BBS})$ . Assuming BBS is SUF-CMA and the DL assumption holds in our group  $\mathbb{G}_1$ ,  $\Pi_{MHAC-BBS}$  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.

Proof. Proof follows from Lemmas 2, 3, 4, 5, and 6.

# 7.1 Correctness of BBS MHAC

**Lemma 2.**  $\Pi_{MHAC-BBS}$  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.**  $\Pi_{MHAC-BBS}$  satisfies presentation unlinkability (Definition 8) in the programmable random oracle model.

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

For what concerns the multi-holder BBS anonymous credential scheme, it being compatible with the BBS anonymous credential scheme [TZ23], we can choose as SimCredPres(pp, pk,  $\tau$ ,  $\{a_i\}_{i \in \text{Rev}}$ , nonce) the same algorithm used to simulate the generation of presentation of a BBS anonymous credential presented in [TZ23] that we recall in Appendix B<sup>16</sup>. 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.  $\hfill \Box$ 

#### 7.3 Unlinkability of private attributes

Lemma 4.  $\Pi_{\text{MHAC-BBS}}$  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  $\mathcal{C}$ .

<sup>&</sup>lt;sup>16</sup> We recall that, together with the public key  $\mathsf{pk}$ , 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].

#### 7.4 Presentation with identifiable abort of BBS MHAC.

**Lemma 5.** Assuming  $\mathcal{H}_{com}$  is a secure commitment scheme,  $\Pi_{MHAC-BBS}$  satisfies presentation with identifiable abort (Definition 10).

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

Proof Sketch. It is easy to see that correctness holds because if W 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  $\mathsf{P}_i$  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  $\mathsf{P}_i$  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,  $\Pi_{\text{MHAC-BBS}}$  satisfies concurrent unforgeability of presentations (Definition 11) against an active static adversary corrupting less than t holders and an honest-but-curious issuer.

We sketch the security proof of Lemma 6. In Appendix F we instantiate the security experiments to the case of the multi-holder BBS anonymous credential as intermediate step in the description of the security proof that we sketch below and we formally carry out in Appendix F.1.

To prove that  $\Pi_{\mathsf{MHAC-BBS}}$  is unforgeable according to the security notion of Definition 11, we show how it is possible to use an adversary  $\mathcal{F}$  of the experiment  $\mathsf{Exp}_{\mathcal{F},\mathsf{BBS}}^{\mathsf{c-uf-pres}}(\kappa)$  as a subroutine of a reduction  $\mathcal{B}$  to the DL assumption or to the SUF-CMA of the BBS signature scheme.

Proof Sketch. Let us note that if an adversary  $\mathcal{F}$  is able to win the experiment with non-negligible probability, it is capable of producing a forgery which, by definition, is associated either with a credential that has never been issued to  $\mathcal{F}$ or associated with the target credential; recall that the adversary is in possession of t-1 shares of the target credential. More precisely, a reduction that rewinds the adversary  $\mathcal{F}$  will end up extracting (from the adversary's proof of knowledge of a BBS credential) a credential that will fall into one of these cases (as we will show). In particular,  $\mathcal{F}$  loses the experiment  $\mathsf{Exp}_{\mathcal{F},\mathsf{BBS}}^{\mathsf{c-uf-pres}}(\kappa)$  if it outputs a forgery which is compatible with any of the full credentials  $\mathcal{F}$  was issued during the training phase.

We show in Appendix F.1 that if the reduction  $\mathcal{B}$  extracts a credential which corresponds to the target credential (case A) it has issued to  $\mathcal{F}$ , then  $\mathcal{B}$  can break the DL assumption. Instead, if  $\mathcal{B}$ , interacting with  $\mathcal{F}$ , extracts a credential which does not correspond to any of the credentials issued during a credential issuance query (case B), then  $\mathcal{B}$  breaks the qSDH assumption since that credential is a forgery of a BBS signature<sup>17</sup>, and the BBS signature has been proven secure under the qSDH assumption in [TZ23]. It is clear that, in general, for MHAC schemes compatible with secure anonymous credential schemes this proving (case B) is trivial. Some details about the techniques used to design a secure presentation protocol are sketched in Section 1.2.

Our reduction can simulate the unforgeability experiment without rewinding the adversary, therefore the reduction  $\mathcal{B}$  both to the DL assumption described in Appendix F.1, Case A, and to the strong unforgeability of the BBS signatures<sup>18</sup>, Case B, can allow the adversary to open concurrent presentation session during the training phase. This guarantees the concurrent security of the multi-holder BBS anonymous credential scheme  $\Pi_{MHAC-BBS}$ .

Acknowledgments. The first author acknowledges support from Eustema S.p.A. through the PhD scholarship.

# 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<sup>+</sup>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.
- 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.

<sup>&</sup>lt;sup>17</sup> Or of a "wildcard" credential as we describe in Appendix B which the issuer never creates and an adversary who can see a polynomial number of BBS signatures can not create (again under the qSDH assumption).

 $<sup>^{18}</sup>$  The strong unforgeability of BBS signatures is proven to hold in [TZ23] under the qSDH assumption

- BS23. Dan Boneh and Victor Shoup. A graduate course in applied cryptography. Draft 0.6, 2023.
- 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.
- CKM23. 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.
- CL01. Jan Camenisch and Anna Lysyanskaya. An efficient system for nontransferable 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.
- Dam02. Ivan Damgård. On  $\sigma$ -protocols. Lecture Notes, University of Aarhus, Department for Computer Science, 84, 2002.
- DKL<sup>+</sup>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<sup>+</sup>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.

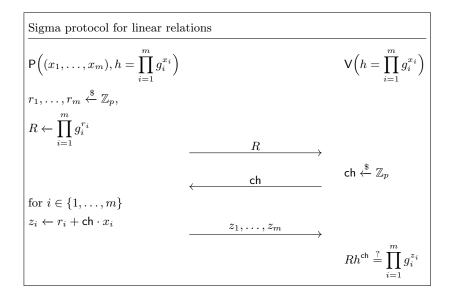
- IOZ14. Yuval Ishai, Rafail Ostrovsky, and Vassilis Zikas. Secure multi-party computation with identifiable abort. In Advances in Cryptology-CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II 34, pages 369–386. Springer, 2014.
- KL07. Jonathan 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  $\sigma$ -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  $pp \leftarrow (\mathbb{G}, p, g_1, \ldots, g_m)$ , 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

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



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

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

# **B** Analysis of BBS Presentation Protocol in [TZ23].

In this section we focus on the security analysis of Protocol 3 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 3. 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  $r \stackrel{\$}{\leftarrow} \mathbb{Z}_p$  and computes  $\overline{A} \leftarrow A^r$   $\overline{B} \leftarrow C(\mathbf{a})^r \overline{A}^{-e} = C(\mathbf{a}')^r \left(\prod_{i \in \mathsf{Hid}} h_i^{ra_i}\right) \overline{A}^{-e}$ 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:  $\mathcal{R}_{\mathsf{LR}}(\mathbb{G}_1, p, C(\mathbf{a}'), \{h_i\}_{i \in \mathsf{Hid}}, \overline{A}) =$  $= \left\{ \left( \underbrace{(x, \{y_j : j \in \mathsf{Hid}\}, z)}_{witness}, \underbrace{\overline{B}}_{statement} : \overline{B} = C(\mathbf{a}')^x \left(\prod_{j \in \mathsf{Hid}} h_j^{y_j} \right) \overline{A}^z \right\} (2)$ 

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  $\overline{B}^{19}$ .

*HVZK.* 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<sup>20</sup>. In fact, a simulator can simulate the signature material randomization phase by sampling  $r \stackrel{\$}{\leftarrow} \mathbb{Z}_p$  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.* For what concerns the special soundness property, observe that the sigma protocol for linear relation is a knowledge sound sigma protocol,

<sup>&</sup>lt;sup>20</sup> 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  $\mathbf{e}(S, X_2) = \mathbf{e}(T, g_2)$ 

Interactive BBS credential presentation of [TZ23] **Prover**  $\mathsf{P}((A, e, \{a_i\}_{i \in \mathsf{Hid}}), (\{a_j\}_{j \in \mathsf{Rev}}))$ Verifier  $V(\{a_j\}_{j\in \mathsf{Rev}})$  $r \xleftarrow{\$} \mathbb{Z}_n, \ \overline{A} \leftarrow A^r, \ \overline{B} \leftarrow C(\mathbf{a})^r \overline{A}^{-e}$  $v_r, \{v_j\}_{j \in \mathsf{Hid}}, v_e \stackrel{\$}{\leftarrow} \mathbb{Z}_p,$  $U \leftarrow C(\mathbf{a}')^{v_r} \Big(\prod_{j \in \mathsf{Hid}} h_j^{v_j}\Big) \overline{A}^{v_e}$  $\xrightarrow{\overline{A},\,\overline{B},\,U}$  $\mathsf{ch} \xleftarrow{\$} \mathbb{Z}_p$ ch  $z_r \leftarrow v_r + \mathsf{ch} \cdot r$  $z_i \leftarrow v_i + \mathsf{ch} \cdot ra_i, \ \forall j \in \mathsf{Hid}$  $z_e \leftarrow v_e + \mathsf{ch} \cdot (-e)$ rsp $\mathsf{rsp} \leftarrow (z_r, \{z_j\}_{j \in \mathsf{Hid}}, z_e)$  $U\overline{B}^{\mathsf{ch}} \stackrel{?}{=} C(\mathbf{a}')^{z_r} \left(\prod_{i \in \mathsf{Hid}} h_j^{z_j}\right) \overline{A}^{z_e}$  $\mathbf{e}(\overline{A}, X_2) \stackrel{?}{=} \mathbf{e}(\overline{B}, g_2)$ 

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

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

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

and also that

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

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 signa-

<sup>&</sup>lt;sup>21</sup> 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  $\overline{B}$ ).

ture (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  $i \in \text{Hid}$  then the extractor can compute  $\overline{B} = (\prod_{i \in \text{Hid}} h_i^{\mu_i}) \overline{A}^{\epsilon}$ . But

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

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  $\overline{A} = (\prod_{i \in \mathsf{Hid}} h_i^{\mu_i})^{\frac{1}{x+e}}$ . We refer to such a tuple as wildcard credential<sup>22</sup>.

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.

$$R(\mathsf{pp}, X_2) = = \left\{ \left( \underbrace{(A, e, \{a_i\}_{i \in \mathsf{Hid}})}_{\text{witness}}, \underbrace{\{a_j\}_{j \in \mathsf{Rev}}}_{\text{statement}} \right) : \mathbf{e}(A, X_2 g_2^e) = \mathbf{e} \left( g_1 \prod_{i \in \mathsf{Hid}} h_i^{a_i} \prod_{j \in \mathsf{Rev}} h_j^{a_j}, g_2 \right) \\ \vee \mathbf{e}(A, X_2 g_2^e) = \mathbf{e} \left( \prod_{i \in \mathsf{Hid}} h_i^{a_i}, g_2 \right) \right\}$$
(5)

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

**Observation 3.** 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  $\mathbf{e}(A, X_2g_2^e) = \mathbf{e}(g_1 \prod_{i \in \mathsf{Hid}} h_i^{a_i} \prod_{j \in \mathsf{Rev}} h_j^{a_j}, g_2)$ . 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 presentation are proofs of possession of a BBS credential.

<sup>&</sup>lt;sup>22</sup> We call it wildcard credential because this credential would allow the holder to present any attribute in the positions  $j \notin \text{Hid}$  setting the exponent of  $C(\mathbf{a}')$  to 0

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*  $((A, e), \{a_i\}_{i \in Hid})$  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 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 3 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.

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

**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  $pk = X_2$ .

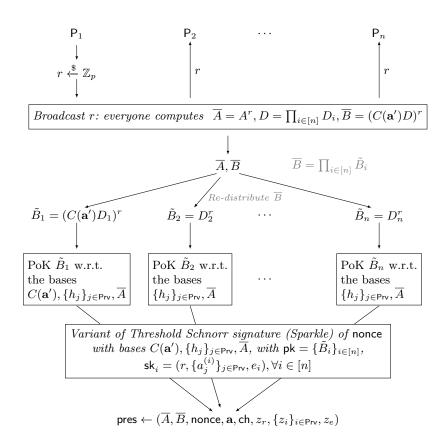
# C BBS MHAC Presentation Protocol Overview

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

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

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

The boxes representing the PoK of  $\tilde{B}_i$ , describes 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.



**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  $\mathsf{P}_1$ .

# 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  $(x, g_2^x), x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ .
- 2.  $\mathcal{A}$  of 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  $s \stackrel{\$}{\leftarrow} \mathbb{Z}_p$  and sends it to  $\mathcal{B}$ .
- 4.  $\mathcal{B}$  generates a BBS credential  $\mathsf{cred} = (A, e)$  with  $e \stackrel{\$}{\leftarrow} \mathbb{Z}_p$  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  $\{\mathsf{P}_i\}_{i\in[n-1]}$ ):

$$- \mathcal{B} \text{ samples, } (a_b^{*(i)}, s^{(i)}, e^{(i)}) \xleftarrow{\$} \mathbb{Z}_p^3, \forall i \in [n-1];$$

- sets 
$$D_i = h_{m-1}^{a_b^{(i)}} A^{-e^{(i)}}, D_n \leftarrow \frac{D}{\prod_{i \in [n-1]} D_i}, \text{ and } e^{(n)} \leftarrow e - \sum_{i \in [n-1]} e^{(i)}$$

$$-$$
 sets for all  $i \in [n-1]$ 

$$\operatorname{cred}_{i} = \left(A, e^{(i)}, \{D_k\}_{k \in [n] \setminus \{i\}}, (\{a_j\}_{j \in \mathsf{Pub}}, a_b^{*(i)}, s^{(i)})\right),$$

and

$$\operatorname{cred}_{n} = \left(A, e^{(n)}, \{D_{k}\}_{k \in [n] \setminus \{n\}}, (\{a_{j}\}_{j \in \mathsf{Pub}}, \bot, \bot)\right).$$

 $-\mathcal{B}$  sends the shares  $\operatorname{cred}_i$  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  $\operatorname{ch}, z_{m-1}^{(n)}, z_m^{(n)}, z_e^{(n)} \stackrel{\$}{\leftarrow} \mathbb{Z}_p$  and setting  $U_n \leftarrow \tilde{B}_n^{-\operatorname{ch}} h_{m-1}^{z_m^{(n)}} h_m^{z_e^{(n)}} \mathcal{B}$  obtains  $(U_n, \operatorname{ch}, z_{m-1}^{(n)}, z_m^{(n)}, z_e^{(n)})$ .

- 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  $\mathcal{H}_{sig}(\mathsf{nonce}, U, \overline{A}, \overline{B}, \{a_i\}_{i \in \mathsf{Rev}}) =$ 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 2), satisfies the definition of presentation with identifiable abort, we construct a wrapper algorithm W for CredPres that satisfies correctness and identifiability. 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), then W aborts with output (abort, j). Otherwise, at the conclusion of the protocol, W outputs  $\bot$ . Note that the only steps of CredPres which could cause an abort are the checks in Steps 5 and 7.

Note that we assume  $\mathcal{H}_{com}$  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  $\bot$ , 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<sub>i</sub> 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.
- $\mathsf{P}_i$  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  $\mathsf{P}_j$ 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.  $\mathsf{P}_i$  outputs (abort, k) in Step 5. It means that  $\mathsf{P}_k$  did not open correctly its commitment, therefore it is malicious.
- There exists j s.t.  $\mathsf{P}_i$  outputs (abort, j) in Step 7. It means that  $\mathsf{P}_k$  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.

$$\begin{split} & \underbrace{\mathsf{Exp}_{\mathcal{A}}^{\mathsf{c}-\mathsf{uf}-\mathsf{pres}}(\kappa):} \\ & \mathsf{pp} \stackrel{\$}{\leftarrow} \mathsf{Pgen}(\kappa) \\ & (\mathsf{pk},\mathsf{sk}) \stackrel{\$}{\leftarrow} \mathsf{KGen}(\mathsf{pp}) \\ & (\mathsf{nonce}^{\star}, \{a_i^{\star}\}_{i \in \mathsf{Rev}^{\star}}, \mathsf{pres}^{\star}) \stackrel{\$}{\leftarrow} \mathcal{A}^{\mathcal{O}_{\mathsf{iss}}, \mathcal{O}_{\mathsf{pres}}}(\mathsf{pp}, \mathsf{pk}) \\ & /\!\!/ \text{ verification of the validity of the forgery} \\ & \mathsf{flag} \leftarrow 1 \\ & \mathsf{if} \left( (\mathsf{nonce}^{\star}, \{a_i^{\star}\}_{i \in \mathsf{Rev}^{\star}}, \mathsf{pres}^{\star}) \in \mathsf{PT} \land \mathsf{pres}^{\star} \neq \bot \right) \lor \left( (\mathsf{nonce}^{\star}, \{a_i^{\star}\}_{i \in \mathsf{Rev}^{\star}}, \bot) \in \mathsf{PT} \right) \\ & \mathsf{flag} \leftarrow 0 \\ & \mathsf{for} \ \mathsf{cred} \in \mathsf{CT} \quad /\!\!/ \ \mathsf{for} \ \mathsf{attributes} \ \{a_i\}_{i \in [m]} \\ & \mathsf{if} \ \{a_i\}_{i \in \mathsf{Rev}^{\star}} = \{a_i^{\star}\}_{i \in \mathsf{Rev}^{\star}} \\ & \mathsf{flag} \leftarrow 0 \\ & \mathsf{return} \left( \mathsf{VfPres}(\mathsf{nonce}^{\star}, \{a_i^{\star}\}_{i \in \mathsf{Rev}}, \mathsf{pres}^{\star}) = 1 \right) \land \mathsf{flag} \end{split}$$

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

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

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

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

- Random oracle queries: the adversary has access to a random oracles  $\mathcal{RO}_{sig}$ ,  $\mathcal{RO}_{com}$ 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  $\kappa$ .
- Credential issuance queries: the oracle  $\mathcal{O}_{iss}$  initialises an empty credential table CT. We describe the two kind of queries, namely queries for a full credentials or for the partial credentials:
  - queries for full credentials:  $\mathcal{O}_{iss}$  takes in input from  $\mathcal{F}$  a set of attributes  $\{a_i\}_{i\in[m]}$  and provides  $\mathcal{F}$  with a credential cred =  $((A, e), \{a_i\}_{i\in[m]})$ , where (A, e) is a BBS signature on  $\{a_i\}_{i\in[m]}$ .<sup>23</sup> Finally the oracle stores in CT a record

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

 $\bullet$  query for the target credential:  ${\cal F}$  sends as input to  ${\cal O}_{iss}$  a tuple

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

then  $\mathcal{O}_{iss}$  generates a credential 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), Pub it has received in input generating the shares

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

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

Finally  $\mathcal{O}_{iss}$  sets targetCred to

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

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

- Presentation queries: the presentation oracle  $\mathcal{O}_{\text{pres}}$  initialises the presentation table PT and has reading access to record targetCred. The adversary can query the presentation oracle  $\mathcal{O}_{\text{pres}}$  for a presentation of the target credential and queries for the presentation of credentials that it was not issued:

<sup>&</sup>lt;sup>23</sup> Note that  $\mathcal{F}$  from a BBS signature can generate a multi-holder BBS credential for any possible parameter (t, n), Pub choice.

• query for the presentation of targetCred:  $\mathcal{F}$  inputs to  $\mathcal{O}_{pres}$  a tuple

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

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

The adversary possibly opens many concurrent sessions of presentation of the target credential, each of them is identified by an identifier ssid.

 query for the presentation of a credential which has never been issued to F: F inputs to O<sub>pres</sub> a tuple (nonce, {a<sub>i</sub>}<sub>i∈Rev</sub>) and O<sub>pres</sub> generates autonomously (as we have described in Experiment 3) a presentation pres for nonce and {a<sub>i</sub>}<sub>i∈Rev</sub> and sends it to F. Finally, O<sub>pres</sub> stores in PT a record

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

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

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

According to Definition 11, a protocol for the presentation of a BBS multiholder anonymous credentials is unforgeable if  $\Pr\left[\mathsf{Exp}_{\mathcal{F},\mathsf{BBS}}^{\mathsf{c-uf-pres}}(\kappa) = 1\right]$  is negligible in the security parameter  $\kappa$ .

### F.1 Unforgeability proof

In this section we prove the unforgeability of the multi-holder BBS presentation protocol described in Protocol 2.

*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 2, 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 cor = [n - 1], therefore the adversary will be provided with the n - 1 shares

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

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

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

- 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

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

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  $\mathcal{RO}_{sig}$  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  $\mathsf{Exp}_{\mathcal{F},\mathsf{BBS}}^{\mathsf{c}-\mathsf{uf}-\mathsf{pres}}(\kappa)$ 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  $\mathsf{Exp}_{\mathcal{B}}^{\mathsf{DL}}(\kappa)$  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 follows: 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  $k = g^x h$ .  $\begin{array}{c} \displaystyle \underline{\mathcal{B}}.\mathsf{Setup}(p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,\mathbf{e},g,h) \\ \hline g_2 \stackrel{\$}{\leftarrow} \mathbb{G}_2 & /\!\!/ \text{ generator for } \mathbb{G}_2 \\ x \stackrel{\$}{\leftarrow} \mathbb{Z}_p & /\!\!/ \text{ BBS issuer secret key} \\ \displaystyle X_2 \leftarrow g_2^x & /\!\!/ \text{ BBS issuer public key} \\ k \leftarrow g^x h & /\!\!/ \text{ note that if } \log_g h = e, \text{ then } g^x h = g^{x+e} \\ \displaystyle \gamma_0,\gamma_1,\ldots,\gamma_m \stackrel{\$}{\leftarrow} \mathbb{Z}_p \\ g_1 \leftarrow k^{\gamma_0} & /\!\!/ \text{ generator for } \mathbb{G}_1 \\ h_i = k^{\gamma_i},\forall i \in [m] \\ \mathsf{pp} \leftarrow (p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,\mathbf{e},g_1,g_2,h_1,\ldots,h_m) \\ \mathsf{HT}_{\mathsf{com}},\mathsf{HT}_{\mathsf{sig}} \leftarrow \emptyset & /\!\!/ \text{ hash tables} \\ \mathsf{CT},\mathsf{PT} \leftarrow \emptyset & /\!\!/ \text{ credential and presentation tables} \\ \displaystyle c_H,c_I,c_P \leftarrow 0 & /\!\!/ \text{ counters for random oracle, issuing and presentation queries} \\ \mathsf{targetCred} \leftarrow \bot & /\!\!/ \text{ the queried partial credential} \\ (\mathsf{pp},X_2) \rightarrow \mathcal{F} \\ \mathsf{return} (\mathsf{pp},X_2,\mathsf{HT}_{\mathsf{com}},\mathsf{HT}_{\mathsf{sig}},\mathsf{CT},\mathsf{PT},c_H,c_I,c_P) \end{array}$ 

Random oracles simulation.  $\mathcal{B}$  can simulate the random oracles  $\mathcal{RO}_{com}$  and  $\mathcal{RO}_{sig}$  programming them using the hash tables  $\mathsf{HT}_{com}, \mathsf{HT}_{sig}$  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  $d \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ , stores in the corresponding hash table the record (m, d) and returns d. The counter  $c_H$  is

$\boxed{\mathcal{B}.\mathcal{R}\mathcal{O}_{com}(m)}$	$\mathcal{B.RO}_{sig}(m)$
if $c_H < q_H \wedge HT_{com}(m) = \bot$	$\mathbf{if} \ c_H < q_H \wedge HT_{sig}(m) = \bot$
$c_H \leftarrow c_H + 1$	$c_H \leftarrow c_H + 1$
$d \xleftarrow{\$} \mathbb{Z}_p$	$d \xleftarrow{\$} \mathbb{Z}_p$
$HT_{com}.add(m,d)$	$HT_{sig}.add(m,d)$
$\mathbf{return} \; HT_{com}(m)$	$\mathbf{return}\ HT_{sig}(m)$

shared among the two interfaces  $\mathcal{B}.\mathcal{RO}_{sig}$  and  $\mathcal{B}.\mathcal{RO}_{com}$ .

*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  $\mathcal{O}_{iss}$  by generating a credential according to the BBS

signing algorithm  $\operatorname{Sign}_{\mathsf{BBS}}(x, \{a_i\}_{i \in [m]}) \xrightarrow{\$} (A, e)$ , and sending it to  $\mathcal{F}$ . Then  $\mathcal{B}$  stores in CT the credential  $\operatorname{cred} \leftarrow (A, e, \{a_i\}_{i \in [m]})$ . In this case the simulation is perfect.

- When  $\mathcal{F}$  sends its only issuance query for the target credential giving in input  $(\{a_i\}_{i\in[m]}, t, n, \mathsf{Pub}, \mathsf{cor}), \mathcal{B}$  simulates the issuance of the target credential so that the value  $e^*$  of the underlying BBS signature  $(A, e^*)$  is actually the discrete logarithm  $\log_g h^{24}$ . To do so,  $\mathcal{B}$  must produce two values  $A, \tilde{A}$  s.t.  $\log_A \tilde{A} = \log_g h$ , 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 below. We recall that for the sake of simplicity we assume that t = n and that  $\mathcal{F}$  always chooses to corrupt the parties in  $\mathsf{cor} = [n-1]$ .

 $<sup>^{24}</sup>$  Note that  ${\cal F}$  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

 $\mathcal{B}$ .IssueCred( $\mathbf{a} = \{a_i\}_{i \in [m]}, t, n, \mathsf{Pub}, \mathsf{cor}\}$ if  $(t, n, cor) = (\bot, \bot, \bot) \land (c_I < q_I)$  // full credential query  $c_I \leftarrow c_I + 1$  $(A, e) \xleftarrow{\$} \mathsf{Sign}_{\mathsf{BBS}}(\mathsf{sk} = x, \{a_i\}_{i \in [m]})$  $\mathsf{cred} \leftarrow ((A, e), \{a_i\}_{i \in [m]})$ CT.add(cred) return cred if  $(t, n, cor) \neq (\bot, \bot, \bot) \land targetCred = \bot // partial credential query$  $\alpha \leftarrow \gamma_0 \sum_{i=1}^m \gamma_i a_i$  $A \leftarrow g^{\alpha} \quad /\!\!/ \text{ if } h = g^{e^{\star}} \text{ then } k = g^{x+e^{\star}} \text{ i.e. } A = (k^{\alpha})^{\frac{1}{x+e^{\star}}} = (C(\mathbf{a}))^{\frac{1}{x+e^{\star}}}$  $\tilde{A} \leftarrow h^{\alpha} \quad /\!\!/ \tilde{A} \text{ would be } A^{e^{\star}}$  $/\!\!/ \text{ Note that } A^{x}\tilde{A} = A^{x+e^{\star}} = A^{x}A^{e^{\star}} = g^{\alpha x}h^{\alpha} = (g^{x}h)^{\alpha} = k^{\alpha} = g_{1}\prod_{i=1}^{m} h_{i}^{\alpha_{i}}$  $\{e^{(i)}\}_{i\in\mathsf{cor}} \xleftarrow{\$} Z_p^{|\mathsf{cor}|}$  $\{a_j^{(i)}\}_{i\in[n]} \xleftarrow{\$} \mathsf{Share}(n,t,a_j), \forall j \in \mathsf{Prv}$  $\tilde{A}_{i} \leftarrow A^{-e^{(i)}}, \forall i \in \operatorname{cor} \quad /\!\!/ \text{ we assume cor} = [n-1]$  $\tilde{A}_{n} \leftarrow \tilde{A}^{-1} \prod_{i \in \operatorname{cor}} \tilde{A}_{i}^{-1} \quad /\!\!/ = A^{-e^{\star}} \prod_{i \in \operatorname{cor}} A^{e^{(i)}}$  $D_i \leftarrow \prod h_j^{a_j^{(i)}} \tilde{A}_i, \forall i \in [n]$  ${\mathsf{cred}_i}_{i \in \mathsf{cor}} \leftarrow ((A, \{e^{(i)}\}_{i \in \mathsf{cor}}, \{D_i\}_{i \in [n]}, (\{a_j\}_{j \in \mathsf{Pub}}, \{a_i^{(i)}\}_{j \in \mathsf{Prv}, i \in \mathsf{cor}}))$  $\mathsf{targetCred} \leftarrow ((A, \{e^{(i)}\}_{i \in \mathsf{cor}}, \{D_i\}_{i \in [n]}, (\{a_j\}_{j \in \mathsf{Pub}}, \{a_i^{(i)}\}_{j \in \mathsf{Prv}, i \in [n]}), t, n, \mathsf{cor})$ // Note that  $e^{(n)}$  is not known to  $\mathcal{B}$ **return** {cred<sub>i</sub>}<sub>i \in [cor]</sub> return  $\perp$ 

The simulation of the issuance of the target credential is perfect as well. The resulting value  $\tilde{A}$  will satisfy  $\tilde{A} = A^{e^*}$ , where  $e^*$  is the solution of the input challenge to  $\mathcal{B}$ , i.e.  $e^* = \log_g h$ . Moreover, the set  $\{e^{(i)}\}_{i \in \text{cor}}$  is indistinguishable from an honest execution since, being |cor| < t they are selected uniformly at random.

Presentation oracle simulation. We distinguish the two cases corresponding to queries for presentation of credentials not issued to  $\mathcal{F}$ , and queries for presentations of the target credential.

- When  $\mathcal{F}$  sends a presentation query for non-issued credentials it inputs the triple (nonce,  $\{a_i\}_{i \in \mathsf{Rev}}$ ) to  $\mathcal{O}_{\mathsf{pres}}$ .  $\mathcal{B}$  can simulate the oracle  $\mathcal{O}_{\mathsf{pres}}$  generating a presentation **pres** by simply generating a BBS credential for the attributes

queried by  $\mathcal{F}$  and other random attributes, and generating a valid presentation from it. Since the presentations are unlinkable, it is not possible to distinguish  $\mathcal{B}$  from the presentation oracle  $\mathcal{O}_{pres}$ .  $\mathcal{B}$  in the generation of the presentation might program the hash table  $\mathsf{HT}_{\mathsf{sig}}$  but it does not have to overwrite it, so the simulation is perfect.

Note that  $\mathcal{B}$  does not have to program the hash table  $\mathsf{HT}_{com}$  since it is not asked to provide  $\mathcal{F}$  with the full transcript associated to a honest generation of the presentation, so  $\mathcal{B}$  generates a presentation as described in Protocol  $3^{25}$ .

Then  $\mathcal{B}$  adds to PT the record (nonce,  $\{a_i\}_{i \in \mathsf{Rev}}$ , pres, T). Then  $\mathcal{B}$  sends pres to  $\mathcal{F}$ 

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

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

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

where  $\mathcal{B}$  does not know the values in  $\{e^{(i)}\}_{[n]\setminus \mathsf{cor}}(=e^{(n)})$  which are set to  $\perp$ since  $e^* = \log_q h$  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  $\mathcal{O}_{pres}$ . 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 and  $a_i, j \in \mathsf{Prv}$ .

The simulation of the execution of Protocol 2 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  $\mathsf{P}_{i}, j \in \mathsf{cor}$ , the value r used to generate A and B.

Every party can compute the values  $\overline{A}, C(\mathbf{a}'), \overline{B}$  as prescribed by the presentation protocol 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:

- B̃<sub>i</sub> = D<sub>i</sub><sup>r</sup> w.r.t. the bases {h<sub>j</sub>}<sub>j∈Prv</sub>, A for each i ≠ j;
   whereas the primary participant P<sub>j</sub> will generate a proof of knowledge of  $\tilde{B}_j = C(a')^r D_j^r \prod_{k \in \mathsf{Hid} \backslash \mathsf{Prv}} h_k^{ra_k}$  w.r.t. the bases  $C(\mathbf{a}'), \{h_j\}_{j \in [m]}, \overline{A}$ .

Recall that every party can compute each other value  $\tilde{B}_i, i \in [n]$ . Recall that the reduction  $\mathcal{B}$ , acting on behalf of  $\mathsf{P}_n$  does not know the value  $e^{(n)}$  but can compute the value  $\tilde{B}_n$  by computing  $\overline{B}\prod_{i\in[n-1]}\tilde{B}_i^{-1}$ .

 $<sup>^{25}</sup>$  Note that in this case  $\mathcal{B}$ , instead of generating a credential and presenting it, could also just simulate the presentation generation, but in this case there would be a negligible chance that the simulation fails.

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  $z_e^{(n)}, \{z_i^{(n)}\}_{i\in\mathsf{Prv}} \stackrel{\$}{\leftarrow} \mathbb{Z}_p$  and a challenge  $\mathsf{ch} \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ . Then  $\mathcal{B}$  sets  $U_n \leftarrow \tilde{B}_n^{-\mathsf{ch}} \left(\prod_{i\in\mathsf{Prv}} h_i^{z_i^{(n)}} \overline{A}^{z_e^{(n)}}\right)$ .

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

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

Event  $bad_1$ . The simulation might fail if the random oracle  $\mathcal{H}_{sig}$  needs to be overwritten and we call this event  $bad_1$ .

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  $z_e^{(n)}, \{z_i^{(n)}\}_{i \in \mathsf{Prv}}$ .

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

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

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

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

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

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

Being  $U \leftarrow \overline{B}^{-ch} C(\mathbf{a}')^{z_r} \prod_{j \in \mathsf{Hid}} h_j^{z_j} \overline{A}^{z_e}$ , according to our reduction simplifications,  $\mathcal{F}$  must have sent a query to  $\mathcal{RO}_{\mathsf{sig}}$  for  $(\{a_k\}_{k \in \mathsf{Rev}}, \mathsf{nonce}, \overline{A}, \overline{B}, U)$ , 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 (nonce,  $\{a_i\}_{i \in \mathsf{Rev}}$ , 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  $((A, e^*), \{a_i\}_{i \in [m]})$  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  $\mathsf{DL}$  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:

$$\begin{aligned} \mathsf{Adv}_{\mathcal{B}}^{\mathsf{DL}}(\kappa) &\geq \\ \frac{\left(1 - \frac{q^2}{p}\right)^2}{q_H} \left(\mathsf{Adv}_{\mathcal{F},\mathsf{BBS}}^{\mathsf{c-uf-pres}_{(A)}}(\kappa)\right)^2 - \frac{\left(1 - \frac{q^2}{p}\right)}{p} \left(\mathsf{Adv}_{\mathcal{F},\mathsf{BBS}}^{\mathsf{c-uf-pres}_{(A)}}(\kappa)\right). \end{aligned}$$
(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 targetCred,  $\mathcal{B}$  extracts  $(A, e^*), \{a_1, \ldots, a_m\}$  such that  $A^{e^*} = \tilde{A}$  according to the notation used in the description of the simulation of  $\mathcal{O}_{iss}$ . Since  $A = g^{\alpha}$  and  $\tilde{A} = h^{\alpha}$ , then we can state that  $h = g^{e^*}$ , 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 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  $\operatorname{Exp}_{\mathcal{F},BBS}^{\mathsf{c-uf-pres}}(\kappa)$  can be used to design an attacker  $\mathcal{B}$  of the strong unforgeability experiment for the BBS signature scheme  $\operatorname{Exp}_{\mathcal{B},BBS}^{\mathsf{suf-cma}}(\kappa)$  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:

$$\begin{split} & \mathcal{B}.\mathsf{Setup}(p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,\mathbf{e},g_1,g_2,h_1,\ldots,h_m,X_2) \\ & \mathsf{pp} \leftarrow (p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,\mathbf{e},g_1,g_2,h_1,\ldots,h_m) \\ & \mathsf{pk} \leftarrow X_2 \\ & \mathsf{HT}_{\mathsf{com}},\mathsf{HT}_{\mathsf{sig}} \leftarrow \emptyset \quad /\!\!/ \text{ hash tables} \\ & \mathsf{CT},\mathsf{PT} \leftarrow \emptyset \quad /\!\!/ \text{ credential and presentation tables} \\ & \mathsf{c}_H,c_I,c_P \leftarrow 0 \quad /\!\!/ \text{ credential and presentation oracle, issuing and presentation queries} \\ & \mathsf{targetCred} \leftarrow \perp \quad /\!\!/ \text{ the queried partial credential} \\ & (\mathsf{pp},X_2) \to \mathcal{F} \\ & \mathsf{return} \ (\mathsf{pp},X_2,\mathsf{HT}_{\mathsf{com}},\mathsf{HT}_{\mathsf{sig}},\mathsf{CT},\mathsf{PT},c_H,c_I,c_P) \end{split}$$

*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  $(\{a_i\}_{i\in[m]}, t, n, \mathsf{Pub}, \mathsf{cor})$  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  $e, \{e^{(i)}\}_{i\in[n]} \stackrel{\$}{\leftarrow} \mathsf{Share}(t, n, e)$  and of  $a_k, \{a_k^{(i)}\}_{i\in[n]} \stackrel{\$}{\leftarrow} \mathsf{Share}(t, n, a_k), \forall k \in \mathsf{Prv}$  and sets targetCred to be equal to

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

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

 $\mathcal{B}$ .IssueCred( $\{a_i\}_{i \in [m]}, t, n, \mathsf{Pub}, \mathsf{cor}$ ) if  $(t, n, cor) = (\bot, \bot, \bot) \land c_I < q_I$  // full credential query  $c_I \leftarrow c_I + 1$  $\{a_i\}_{i \in m} \to \mathcal{O}_{SIGN} \quad /\!\!/ \text{ send a sign query to the BBS signing oracle.}$  $(A, e) \leftarrow \mathcal{O}_{\mathsf{SIGN}}$  $\mathsf{cred} \leftarrow ((A, e), \{a_i\}_{i \in [m]})$ CT.add(cred) return cred if  $(t, n, cor) \neq (\bot, \bot, \bot) \land targetCred = \bot // partial credential query$  $\{a_i\}_{i \in m} \rightarrow \mathcal{O}_{\mathsf{SIGN}} \quad /\!\!/ \text{ send a sign query to the BBS signing oracle.}$  $(A, e) \leftarrow \mathcal{O}_{SIGN}$  $\{e^{(i)}\}_{i\in[n]} \xleftarrow{\$} \mathsf{Share}(t, n, e)$  $\{a_k^{(i)}\}_{i \in [n]} \xleftarrow{\$} \mathsf{Share}(t, n, a_k), \forall k \in \mathsf{Prv}$  $\{\mathsf{cred}_i\}_{i\in\mathsf{cor}} \leftarrow (A, \{D_i\}_{i\in[n]}, \{e^{(i)}\}_{i\in\mathsf{cor}}, (\{a_k\}_{k\in\mathsf{Pub}}, \{a_k^{(i)}\}_{k\in\mathsf{Prv}, i\in\mathsf{cor}}))$ // In the reduction we will consider t = n and cor = [n - 1] $\mathsf{targetCred} \leftarrow (A, \{e^{(i)}\}_{i \in [n]}, \{D_i\}_{i \in [n]}, (\{a_j\}_{j \in \mathsf{Pub}}, \{a_i^{(i)}\}_{j \in \mathsf{Prv}, i \in \mathsf{cor}}), t, n, \mathsf{cor})$ // Note that  $\{e^{(i)}\}_{i\in[n]\setminus cor}$  are known to  $\mathcal{B}$ **return** {cred<sub>i</sub>}<sub>i \in [cor]</sub> return  $\perp$ 

Presentation oracle simulation. Again we distinguish the simulation of queries for presentations of credentials never issued to  $\mathcal{F}$ , and presentations of the target credential.

- The simulation of the presentation of credentials that have never been issued to  $\mathcal{F}$  works differently compared to Case A, where the reduction knows the secret key of the issuer. In this case the  $\mathcal{B}$  can not generate credentials on its own, but instead it runs the BBS presentation simulator described by Tessaro and Zhu [TZ23] that we recall in Appendix B, and generates the queried presentation.

Event bad<sub>2</sub>. In this case, the simulation fails if  $HT_{sig}(nonce, U, \overline{A}, \overline{B}, \{a_k\}_{k \in Rev})$  must be overwritten, and we call this event bad<sub>2</sub>.

- For what concerns 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 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  $\mathsf{cred}^* = ((A, e), \{a_i\}_{i \in [m]}).$ 

In this section we consider the case in which the extracted credential is associated to a credential never issued to  $\mathcal{F}$ , therefore  $\operatorname{cred}^* \notin \operatorname{CT}$ , 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  $\operatorname{Exp}_{\mathcal{F},\mathsf{BBS}}^{\mathsf{suf}-\mathsf{cma}}(\kappa)$  the forgery  $((A, e), \{a_i\}_{i \in [m]})$ , which is a valid forgery for the strong unforgeability experiment for the BBS signature.

We evaluate the advantage  $\operatorname{Adv}_{\mathcal{B}, \operatorname{BBS}}^{\operatorname{suf-cma}}(\kappa)$  of the reduction  $\mathcal{B}$  in winning the strong unforgeability experiment for BBS signatures [TZ23] as a function of the advantage  $\operatorname{Adv}_{\mathcal{F}, \operatorname{BBS}}^{\operatorname{c-uf-pres}(B)}(\kappa)$  of the adversary  $\mathcal{F}$  of the presentation unforgeability experiment in Appendix G.2 where we show that:

$$\operatorname{Adv}_{\mathcal{B},\mathsf{BBS}}^{\mathsf{suf-cma}}(\kappa) \geq \frac{\left(1 - \frac{q^2}{p}\right)^2}{q_H} \left(\operatorname{Adv}_{\mathcal{F},\mathsf{BBS}}^{\mathsf{c-uf-pres}_{(B)}}(\kappa)\right)^2 - \frac{\left(1 - \frac{q^2}{p}\right)}{p} \left(\operatorname{Adv}_{\mathcal{F},\mathsf{BBS}}^{\mathsf{c-uf-pres}_{(B)}}(\kappa)\right). \quad (7)$$

*Remark 8.* 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 unforgeability 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  $bad_1$  occurs as a consequence of the queries of  $\mathcal{F}$ . The events happen if the hash table  $\mathsf{HT}_{sig}$  is overwritten during the simulation.

Being  $i \in [q_P]$  the index representing the *i*-th presentation query performed by  $\mathcal{F}$ , and by  $\mathsf{bad}_1^{(i)}$  the event that  $\mathsf{bad}_1$  happens during the *i*-th presentation query, we can upper-bound the probability that the event  $\mathsf{bad}_1^{(i)}$  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

(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  $\mathsf{HT}_{sig}$  has at most  $q_H + q_P$  elements during the simulation execution, therefore the probability that event  $bad_1^{(i)}$  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

$$\Pr[\mathcal{B} \text{ simulates}] = \Pr\left[\bigwedge_{i \in [q_P]} \neg \mathsf{bad}_1^{(i)}\right] \ge \prod_{i \in [q_P]} \left(1 - \frac{q}{p}\right) \ge \left(1 - \frac{q}{p}\right)^q \ge 1 - \left(\frac{q^2}{p}\right) \quad (8)$$

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,

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

since writing explicitly the products, one obtains

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

which holds for  $p \gg q$ .

Therefore, assuming that q is odd <sup>26</sup>

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

<sup>&</sup>lt;sup>26</sup> If q is even, we should add an extra *positive* element  $\left(\frac{2q}{p}\right)^{3q}$  which makes the relation hold an extra positive element  $\left(\frac{2q}{p}\right)^{3q}$ hold anyways.

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

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

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

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

- $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{DL}}(\kappa)$  as the probability that  $\mathcal{B}$  interacting with  $\mathcal{F}$  wins the  $\mathsf{DL}$  experi-
- ment;  $\operatorname{Adv}_{\mathcal{F}, \operatorname{BBS}}^{\operatorname{c-uf-pres}_{(A)}}(\kappa) = \Pr\left[\mathcal{F}^{\mathcal{B}} \text{ wins } \operatorname{c-uf-pres}_{(A)} | \mathcal{B} \text{ simulates}\right]$  as the probability that  $\mathcal{F}$  wins the unforgeability of presentation forging the target creation for the solution of the solu dential 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  $\mathsf{Adv}_{\mathcal{F}}^{\mathsf{c-uf-pres}_{(A)}}(\kappa)$ );  $-\mathcal{B}$  does not fail the simulation of the experiment.

This algorithm succeeds with probability

$$\epsilon_{A} = \Pr\left[\mathcal{F}^{\mathcal{B}} \text{ wins } \mathsf{c} - \mathsf{u}\mathsf{f} - \mathsf{pres}_{(A)} \land \mathcal{B} \text{ simulates}\right] = \Pr[\mathcal{B} \text{ simulates}] \Pr\left[\mathcal{F}^{\mathcal{B}} \text{ wins } \mathsf{c} - \mathsf{u}\mathsf{f} - \mathsf{pres}_{(A)} | \mathcal{B} \text{ simulates}\right] = \Pr[\mathcal{B} \text{ simulates}] \mathsf{Adv}_{\mathcal{F},\mathsf{BBS}}^{\mathsf{c}-\mathsf{u}\mathsf{f}-\mathsf{pres}_{(A)}}(\kappa) \geq \left(1 - \frac{q^{2}}{p}\right) \mathsf{Adv}_{\mathcal{F},\mathsf{BBS}}^{\mathsf{c}-\mathsf{u}\mathsf{f}-\mathsf{pres}_{(A)}}(\kappa) \quad (10)$$

Therefore, by the Generalized Forking lemma we have

$$\operatorname{Adv}_{\mathcal{B}}^{\mathsf{DL}}(\kappa) \geq \epsilon_{A} \left(\frac{\epsilon_{A}}{q_{H}} - \frac{1}{p}\right) \geq \frac{\left(1 - \frac{q^{2}}{p}\right)^{2}}{q_{H}} \left(\operatorname{Adv}_{\mathcal{F},\mathsf{BBS}}^{\mathsf{c-uf-pres}_{(A)}}(\kappa)\right)^{2} - \frac{\left(1 - \frac{q^{2}}{p}\right)}{p} \left(\operatorname{Adv}_{\mathcal{F},\mathsf{BBS}}^{\mathsf{c-uf-pres}_{(A)}}(\kappa)\right).$$
(11)

#### 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.

We compute the probability of failure of the simulation which happens only according to event bad<sub>2</sub>, which happens with probability  $\Pr[\mathsf{bad}_2^{(i)}] \leq \frac{q_H + q_P}{p} = \frac{q}{p}, \forall i \in [q_P].$ 

Therefore, the simulation does not fail at the *i*-th presentation query with probability  $1 - \frac{q}{p}$  and it never fails with probability

$$\Pr[\mathcal{B} \text{ simulates}] \ge \left(1 - \frac{q}{p}\right)^{q_P} \ge \left(1 - \frac{q}{p}\right)^q \ge 1 - \frac{q^2}{p}.$$

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

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

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 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

$$\epsilon_{B} = \Pr\left[\mathcal{F}^{\mathcal{B}} \text{ wins } \mathsf{c} - \mathsf{u}\mathsf{f} - \mathsf{pres}_{(B)} \land \mathcal{B} \text{ simulates}\right] = \Pr[\mathcal{B} \text{ simulates}] \Pr\left[\mathcal{F}^{\mathcal{B}} \text{ wins } \mathsf{c} - \mathsf{u}\mathsf{f} - \mathsf{pres}_{(B)} | \mathcal{B} \text{ simulates}\right] = \Pr[\mathcal{B} \text{ simulates}] \mathsf{Adv}_{\mathcal{F},\mathsf{BBS}}^{\mathsf{c}-\mathsf{u}\mathsf{f}-\mathsf{pres}_{(B)}}(\kappa) \geq \left(1 - \frac{q^{2}}{p}\right) \mathsf{Adv}_{\mathcal{F},\mathsf{BBS}}^{\mathsf{c}-\mathsf{u}\mathsf{f}-\mathsf{pres}_{(B)}}(\kappa) \quad (12)$$

By applying the Generalised Forking Lemma [BN06] we obtain

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

# 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 CredIss<sub>BBS</sub>). 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  $\operatorname{cred}_i = (A, e^{(i)}, \sigma_i, D_i, (\{a_j\}_{j \in \mathsf{Pub}}, \{a_j^{(i)}\}_{j \in \mathsf{Prv}}))$ , 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  $\sigma_i$  is a digital signature  $\sigma_i = \operatorname{Sign}(h||D_i, \mathsf{sk})$  ( $D_i = (\prod_{j \in \mathsf{Prv}} h_j^{a_j^{(i)}})A^{-e^{(i)}}$ ) created using a secret key  $\mathsf{sk}$  possessed by the issuer. Then, in the third step of the protocol the parties taking part to the presentation protocol execution broadcast their  $\mathsf{com}_i$  together with  $D_i$  and the signature  $\sigma_i$  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  $e^{(i)}, \{a_j^{(i)}\}_{j \in \mathsf{Prv}} \, s.t. \, D_i = (\prod_{j \in \mathsf{Prv}} h_j^{a_j^{(i)}})A^{-e^{(i)}}$ .

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  $(sk_h, pk_h)$ , signs the values  $\tilde{A}_i$  using a secret key  $sk_h$  obtaining  $\sigma_i$ , and sends the pair  $(\sigma_i, pk_h)$  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 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  $a^*$  computed using the randomness  $s \notin \mathbb{Z}_p$  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 want to distribute the private attribute  $a^*$  performs the following operations.

- 1. The dealer computes a Pedersen commitment to  $a^*$  sampling  $s \stackrel{\$}{\leftarrow} \mathbb{Z}_p$  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

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

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  $\mathsf{P}_i$  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  $\mathsf{P}_i$  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 knows also 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) execute the following operations:

- P<sub>i</sub> verifies the shares received by the other parties, and if one does not verify,  $\mathsf{P}_i$  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.