Issuer Hiding for BBS-Based Anonymous Credentials

Jonathan Katz[1] and Marek Sefranek[2]

[1] Google, USA
jkatz2@gmail.com
[2] TU Wien, Austria
marek.sefranek@tuwien.ac.at

Abstract. *Anonymous-credential schemes* allow users to obtain credentials on various attributes, and then use those credentials to give unlinkable proofs about the values of some attributes without leaking anything about others. They have recently received interest from companies including Google, Apple, and Cloudflare, and are being actively evaluated both at the IETF and in the EU. Anonymous credentials based on BBS signatures are a leading candidate for standardization.

In some natural applications of anonymous credentials, it is beneficial to hide even the *issuer* of a credential, beyond revealing the fact that the issuer is in some pre-determined set specified by a verifier. Sanders and Traoré recently showed a construction of such *issuer-hiding* anonymous credentials based on the Pointcheval–Sanders signature scheme.

In this work we show how to achieve issuer hiding for BBS-based anonymous credentials. Our construction satisfies a notion of everlasting issuer-hiding anonymity, and is unforgeable in the generic group model. It can be integrated into existing standards, and has several efficiency advantages compared to prior work.

Keywords: BBS signatures · anonymous credentials

1 Introduction

Anonymous credentials are a privacy-preserving mechanism allowing users to obtain credentials certifying various attributes, and then use those credentials to give unlinkable proofs about the values of some of those attributes while hiding others. Typical usage of anonymous credentials considers an *issuer* along with multiple *users* and *verifiers*.[1] In a system supporting n attributes, a user interacts with an issuer to obtain a credential on a vector of attributes $(m_1, \ldots, m_n)$; the issuer generates the credential using a private key it holds. (Verification of the attributes by the issuer is assumed to be done out of band.) At some later point in time, that user can prove to a verifier—who knows the public

[1] In the so-called *keyed-verification* setting [9,22], the issuer is also the (only) verifier. We consider the publicly verifiable setting here.

key of the issuer—that it possesses a credential on some attributes $(m_i)_{i \in \mathcal{D}}$, where $\mathcal{D}$ is the set of *disclosed attributes*, without revealing anything about the *hidden attributes* $(m_i)_{i \notin \mathcal{D}}$. Anonymity additionally requires that such proofs are unlinkable with each other as well as with the credential-issuance protocol, even if the verifier and issuer collude. Anonymous credentials must also satisfy unforgeability; roughly, it should be infeasible for a user to generate a valid proof for some set of disclosed attributes $(m_i)_{i \in \mathcal{D}}$ unless that user has at some point obtained a credential on a vector of attributes containing exactly those values.

A prototypical proposed application of anonymous credentials is national-level identity verification. To take a somewhat simplified example, a user might obtain a credential from their country's government with the user's name encoded as the first attribute m_1, the user's address encoded as the second attribute m_2, and the user's age encoded as the third attribute m_3. It would then be possible for that user to give a proof about their claimed age without revealing anything about their name or address. Applications of this sort are being explored by ISO/IEC [17], the Verifiable Credentials Working Group of the W3C [28], as part of the European Digital Identity (EUDI) Wallet Solution [14], and in several drafts under consideration at the IETF [19,18,30,25,29].

Although anonymous credentials can be used to disclose some attributes while hiding others, there are settings in which usage of an anonymous credential can itself reveal extraneous information. Consider, in particular, a hypothetical scenario involving the EUDI. It is unlikely that there will be a single issuer for the entire European Union; more likely, each country (or even different administrative regions within a country) will act as an issuer, each with its own public/private key. While it is possible to use standard anonymous credentials in this case, doing so would require a user to inform a verifier which public key to use for verification; that, in turn, would reveal the user's home country (or possibly even more fine-grained information about where they reside), information the user is not necessarily required to disclose. Note that users of the system may not even realize that such information leakage is occurring since users' home countries are not encoded as an explicit attribute in the scheme.

An *issuer-hiding* anonymous-credential scheme would prevent, or at least mitigate, such leakage. Such a scheme would allow a verifier to specify a *set* $\mathcal{P}$ of issuers' public keys; a user holding a credential issued by any issuer whose public key is in that set would be able to generate a valid proof for that verifier, without revealing anything beyond the fact that the user has a credential (for the disclosed attributes) issued by some issuer whose key is in that set.

Several recent works [2,3,6,11,13,21,24,10] have identified this problem, and have constructed issuer-hiding schemes. A natural way to construct such a scheme is to use an OR-proof; this leads to proofs of size linear [2,11] or logarithmic [13] in the number of issuers $|\mathcal{P}|$. A slightly different approach was taken by Bobolz et al. [3]. In their scheme a verifier signs each public key in $\mathcal{P}$, and a user then proves (roughly) that it holds a credential that is valid under a signed public key. This results in constant-size—but concretely large—proofs.

Very recently, Sanders and Traoré [24] showed a construction of an efficient issuer-hiding scheme with constant-size proofs. Rather than relying on a "generic" zero-knowledge approach, they use algebraic techniques to achieve issuer hiding for credentials based on Pointcheval–Sanders (PS) signatures [23], which are a publicly verifiable version of the keyed-verification credentials of Chase et al. [9]. In their scheme, a verifier who is willing to accept credentials issued by any issuer in some set $\mathcal{P}$ generates a (public) *policy key* $\mathsf{pk}_{\mathcal{P}}$ based on $\mathcal{P}$, along with an associated (private) verification key $\mathsf{sk}_{\mathcal{P}}$. A user who holds a credential issued by some issuer in $\mathcal{P}$ can then use $\mathsf{pk}_{\mathcal{P}}$ to generate a proof using that credential; that proof can be verified using $\mathsf{sk}_{\mathcal{P}}$. (Note that although a secret verification key is used for verification, this is still a publicly verifiable setting since anyone can create a verification key using only the public keys of the issuers.)

1.1 Our Contributions

We adapt the work of Sanders and Traoré and achieve issuer hiding for anonymous credentials based on BBS signatures [5,8,1,7,27]. We incorporate issuer hiding without modifying the underlying BBS issuance protocol at all. This is important since several schemes related to BBS-based anonymous credentials are standardized or being considered for standardization [17,19,30,25,18].

Both the Sanders–Traoré scheme and our scheme rely on groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime order p with a bilinear map $\mathsf{e}\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, and analyze security in the generic group model (GGM). Similar to the efficiency advantages[2] of BBS-based anonymous credentials as compared to PS-based anonymous credentials, our issuer-hiding scheme is more efficient—both asymptotically and concretely—than the Sanders–Traoré scheme in several respects. In particular:

- In our scheme, issuers' public keys are constant size, while in the Sanders–Traoré scheme they have size linear in the number of attributes n.
- Policy keys in our scheme have size $O(\ell)$, where ℓ is the size of the set $\mathcal{P}$ of allowed issuers. In the Sanders–Traoré scheme, policy keys have size $O(n \cdot \ell)$.

We provide a more concrete comparison in Tables 1 and 2. In Table 1 we compare the sizes of public keys, credentials, policy keys, and proofs. Note that elements of $\mathbb{Z}_p$ are shorter than elements of $\mathbb{G}_1$, which in turn are shorter than elements in $\mathbb{G}_2$, e.g., for the BLS12-381 curve (assuming point compression is used), elements of $\mathbb{Z}_p$ are 32 bytes, elements of $\mathbb{G}_1$ are 48 bytes, and elements of $\mathbb{G}_2$ are 96 bytes. Our scheme has significantly shorter public keys and policy keys than the Sanders–Traoré scheme, with only slightly larger proofs.

In Table 2 we compare the computational complexities of Show and Vrfy. The numbers here should be understood as rough approximations since we only count group operations and do not take into account preprocessing or possible optimizations like multi-exponentiation algorithms that could be used. Here, too,

[2] We refer to a recent survey [22] for further comparison between BBS- and PS-based credentials. (Although that survey focuses on keyed verification, much of the discussion carries over to the publicly verifiable setting.)

Table 1. Sizes in terms of elements in $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{Z}_p$ assuming one disclosed attribute, where n is the number of attributes and ℓ is the size of the issuer set.

Scheme	Public key	Credential	Policy key	Proof
[24]	$n\,\mathbb{G}_2$	$2\,\mathbb{G}_1$	$(2n\ell{+}n{+}1)\,\mathbb{G}_2 + (n{+}2)\,\mathbb{Z}_p$	$2\,\mathbb{G}_1 + 1\,\mathbb{G}_2 + n\,\mathbb{Z}_p$
Here	$1\,\mathbb{G}_2$	$1\,\mathbb{G}_1 + 1\,\mathbb{Z}_p$	$(\ell{+}1)\,\mathbb{G}_2 + 3\,\mathbb{Z}_p$	$3\,\mathbb{G}_1 + 1\,\mathbb{G}_2 + (n{+}3)\,\mathbb{Z}_p$

Table 2. Computational complexities in terms of pairing operations and exponentiations in $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ assuming one disclosed attribute, where n, ℓ are as in Table 1.

Scheme	Proof generation	Verification
[24]	$3\,\mathsf{exp}_1 + 2n\,\mathsf{exp}_2 + 1\,\mathsf{pair}$	$(n{+}1)\,\mathsf{exp}_2 + 1\,\mathsf{exp}_T + 3\,\mathsf{pair}$
Here	$(2n{+}6)\,\mathsf{exp}_1 + 1\,\mathsf{exp}_2$	$(n{+}5)\,\mathsf{exp}_1 + 2\,\mathsf{exp}_2 + 2\,\mathsf{pair}$

we note that exponentiations in $\mathbb{G}_1$ are fastest, followed by exponentiations in $\mathbb{G}_2$, then exponentiations in $\mathbb{G}_T$, and finally pairings are the slowest. As estimated by Sanders and Traoré [24] for the BLS12-381 curve, compared to exponentiations in $\mathbb{G}_1$, exponentiations in $\mathbb{G}_2$ are 1.9× slower, exponentiations in $\mathbb{G}_T$ are 3.1× slower, and pairings are 9.3× slower. Based on these numbers and Table 2, we expect our scheme to be noticeably faster than the Sanders–Traoré scheme.

1.2 Overview of Techniques

We begin by reviewing BBS-based anonymous credentials. (See Section 2.3 for a formal specification.) The public key of an issuer is $\mathsf{pk} = g_2^x$ for a secret key x. A credential on attributes $\boldsymbol{m} = (m_i)$ is formed by choosing a random exponent e and computing $A = B^{1/(x+e)}$, where $B = g_1 \prod_i h_i^{m_i}$ and the h_i are public, random generators; the credential is (A, e). At a high level—and omitting several details—a user proves possession of a credential (A, e) on attributes $\boldsymbol{m}$ (for simplicity, we assume here all attributes are disclosed) with respect to pk by letting $A' \coloneqq A^r$ be a re-randomization of A, setting $\bar{A} \coloneqq (A')^{-e} B^r$, and then sending $A', \bar{A}$ along with a zero-knowledge (ZK) proof that $\bar{A} = (A')^{-e} B^r$ for some e, r. Besides verifying the ZK proof, verification checks that $\mathsf{e}(A', \mathsf{pk}) = \mathsf{e}(\bar{A}, g_2)$.

Sanders and Traoré [24] show a technique for adding issuer hiding to PS-based anonymous credentials; we show here how to apply their ideas to the case of BBS-based anonymous credentials. Assume now that a user wants to prove possession of a credential (A, e) on attributes $\boldsymbol{m}$ with respect to some key in a *set* $\mathcal{P} = \{\mathsf{pk}_i\}_{i \in [\ell]}$ of ℓ public keys, without revealing which one. Let $\mathsf{pk}_{i^*} \in \mathcal{P}$ be the public key for which the user's credential is valid. A natural idea is to use the product $P = \prod_i \mathsf{pk}_i$ of all the public keys in place of pk in the verification equation. Of course, if we do only this then verification will not succeed. To fix this, we could have the user additionally send a correction term $\tilde{\sigma} = \prod_{i \neq i^*} \mathsf{pk}_i$, so the verifier can check

$$\mathsf{e}(A',\, \tilde{\sigma}^{-1} \cdot P) \stackrel{?}{=} \mathsf{e}(\bar{A}, g_2),$$

but now $\tilde{\sigma}$ reveals pk_{i^*}. This can be addressed by having the user blind $\tilde{\sigma}$, i.e., it now computes $\tilde{\sigma} := g_2^t \cdot \prod_{i \neq i^*} \mathsf{pk}_i$ for uniform t. To ensure that verification succeeds, the user must also adjust $\bar{A}$ by computing it as $\bar{A} := (A')^{-e-t} B^r$.

As described, the scheme is issuer-hiding, but it is no longer unforgeable! In particular, a user could simply set $\tilde{\sigma} = P$ to cancel all the public keys, and then claim possession of a credential on any attributes it likes. To prevent this, we need a way to ensure that the user constructs $\tilde{\sigma}$ using exactly $\ell - 1$ of the public keys in $\mathcal{P}$. Toward this end, the verifier computes a *policy key* $\mathsf{pk}_{\mathcal{P}}$ associated with $\mathcal{P}$ for the user to use when generating their proof. The policy key includes $S = g_2^a$ and $\{T_i = (g_2^b \cdot \mathsf{pk}_i)^a\}_{i \in [\ell]}$ for uniform a, b that are kept private and that will be used for verification. The user is now supposed to compute

$$\tilde{\sigma} := S^t \cdot \textstyle\prod_{i \neq i^*} T_i = g_2^{t \cdot a} \cdot g_2^{(\ell-1) \cdot b \cdot a} \cdot \left(\prod_{i \neq i^*} \mathsf{pk}_i \right)^a,$$

and verification checks that

$$\mathsf{e}\left(A',\ \tilde{\sigma}^{-1/a} \cdot g_2^{(\ell-1) \cdot b} \cdot \textstyle\prod_{i \in [\ell]} \mathsf{pk}_i \right) \stackrel{?}{=} \mathsf{e}(\bar{A}, g_2).$$

Intuitively, incorporating $\tilde{\sigma}^{-1/a}$ into the verification equation forces the user to construct $\tilde{\sigma}$ using the provided values $S, \{T_i\}$, while multiplying by $g_2^{(\ell-1) \cdot b}$ forces the user to use *exactly* $\ell - 1$ of the $\{T_i\}$ when constructing $\tilde{\sigma}$.

2 Preliminaries

We use ":=" for deterministic assignment, and "$\leftarrow$" for the output of a randomized algorithm or uniform selection from a set. We work in a concrete-security setting and thus do not have an explicit security parameter. (Nevertheless, it is immediate that our scheme could be proven secure asymptotically.) In particular, we assume fixed groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime order p with generators g_1, g_2, g_T, respectively, along with a bilinear map $\mathsf{e}\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. We analyze our scheme in the *generic group model* (GGM) for pairing-based groups [26,20,4]. In this model, algorithms are provided with oracles for computing group operations in each of the three groups, as well as for computing the pairing operation.

2.1 Non-Interactive Proofs of Knowledge

We rely on non-interactive zero-knowledge proofs of knowledge (ZKPoKs) for relations involving group elements. For simplicity we assume perfect completeness and perfect zero knowledge; however, these could be relaxed to statistical notions without significantly affecting our results. For issuer-hiding anonymity we require the ZKPoKs to satisfy (computational) soundness only; for unforgeability we require the stronger notion of (tag-based) simulation extractability [12,16,15], which implies that a knowledge extractor can compute a witness from an adversary who outputs a valid proof, even if that adversary is given multiple simulated proofs. In our construction and proofs of security, we abstract out the details of

the ZKPoKs since they distract from the main ideas; also, the ZKPoKs could in principle be instantiated in different ways. In Appendix A we describe suitable ZKPoKs based on standard Σ-protocols using the Fiat–Shamir transform that could be used in our scheme.

2.2 Anonymous Credentials

We first review definitions for standard anonymous credentials, and extend them to support issuer hiding in the following section. In an anonymous-credential scheme there are four main algorithms. The key-generation algorithm KeyGen is run by an issuer to generate a public/private key pair $(\mathsf{pk}, \mathsf{sk})$. The issuance algorithm Issue is run by an issuer to generate a credential σ for a user with attribute vector $(m_1, \ldots, m_n)$. Verification of those attributes (e.g., verifying that a user is the age they claim) is done out of band. We assume n is fixed by the scheme itself, and that all parties in the system agree on the semantics of the attributes. A user with a credential σ for attribute vector $(m_1, \ldots, m_n)$ can run the Show algorithm to selectively disclose some subset $\mathcal{D} \subseteq [n]$ of those attributes to a verifier (who holds the public key of the issuer) while hiding the rest. More formally, the user generates a proof that they hold a credential on some attribute vector matching the disclosed attributes. To prevent replay attacks, proofs are bound to some *context* ctx (see [3]) that might incorporate the current date/time, the verifier's identity, and/or a challenge from the verifier among other things; we model this by having the Show algorithm take ctx as input. Finally, the Vrfy algorithm is used to verify proofs.

We provide the formal definition below. We include a Setup algorithm for generating public parameters; while in general it may be possible for the issuer itself to run this algorithm (and include the output as part of its public key), for issuer hiding we will need to assume that all issuers share the same public parameters. Our definition also includes an algorithm $\mathsf{VrfyCred}$ to verify a credential; this allows a user to check that an issuer behaved honestly.

Definition 1 (Anonymous Credentials). *An anonymous-credential scheme consists of the following algorithms:*

- Setup *outputs public parameters* params*. We assume these are implicitly given as input to all algorithms below.*
- KeyGen *outputs a public/private key pair* $(\mathsf{pk}, \mathsf{sk})$*.*
- Issue *takes as input a secret key* sk *and attributes* $m_1, \ldots, m_n \in \mathbb{Z}_p$*. It outputs a credential* σ*.*
- $\mathsf{VrfyCred}$ *takes as input a public key* pk*, attributes* $m_1, \ldots, m_n \in \mathbb{Z}_p$*, and a credential* σ*. It outputs a bit.*
- Show *takes as input* pk*, a disclosure set* $\mathcal{D} \subseteq [n]$*, attributes* $(m_i)_{i \in [n]}$*, context* ctx*, and a credential* σ*. It outputs a proof* π*.*
- Vrfy *takes as input a public key* pk*, a disclosure set* $\mathcal{D} \subseteq [n]$*, attributes* $(m_i)_{i \in \mathcal{D}}$*, context* ctx*, and a proof* π*. It outputs a bit.*

Correctness requires the following to hold:

1. *For all* $\boldsymbol{m} = (m_1, \ldots, m_n) \in \mathbb{Z}_p^n$*, we have*

$$\Pr\left[\begin{array}{c}\mathsf{params} \leftarrow \mathsf{Setup}; (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}; \\ \sigma \leftarrow \mathsf{Issue}_{\mathsf{sk}}(m_1, \ldots, m_n)\end{array} : \mathsf{VrfyCred}_{\mathsf{pk}}(\boldsymbol{m}, \sigma) \neq 1\right] \leq 1/p.$$

2. *For all* $\boldsymbol{m}$*, all* params *output by* Setup*, any* $(\mathsf{pk}, \mathsf{sk})$ *output by* KeyGen*, any* $\mathcal{D} \subseteq [n]$*, any* σ *with* $\mathsf{VrfyCred}_{\mathsf{pk}}(\boldsymbol{m}, \sigma) = 1$*, any* ctx*, and all* π *output by* $\mathsf{Show}_{\mathsf{pk}}(\mathcal{D}, \boldsymbol{m}, \mathsf{ctx}, \sigma)$*, we have* $\mathsf{Vrfy}_{\mathsf{pk}}(\mathcal{D}, (m_i)_{i\in\mathcal{D}}, \mathsf{ctx}, \pi) = 1$*.*

Anonymous credentials must satisfy two standard security requirements: anonymity and unforgeability. Anonymity is formalized by the following; note that we separate the adversary into two parts in order to obtain a more fine-grained definition. (See the discussion in Section 5.1.)

Definition 2 (Anonymity). *Consider the following experiment involving an adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *and scheme* Π*:*

1. Setup *is run to generate* params*.*
2. $\mathcal{A}_1(\mathsf{params})$ *outputs* $(\mathsf{pk}, \mathcal{D}, \mathsf{ctx}, \boldsymbol{m}_0 = (m_{0,i})_{i\in[n]}, \sigma_0, \boldsymbol{m}_1 = (m_{1,i})_{i\in[n]}, \sigma_1)$ *and state* st *where:*
   - $\mathsf{VrfyCred}_{\mathsf{pk}}(\boldsymbol{m}_0, \sigma_0) = \mathsf{VrfyCred}_{\mathsf{pk}}(\boldsymbol{m}_1, \sigma_1) = 1$ *and*
   - $m_{0,i} = m_{1,i}$ *for all* $i \in \mathcal{D}$*.*
3. *Choose* $b \leftarrow \{0, 1\}$ *and compute* $\pi \leftarrow \mathsf{Show}_{\mathsf{pk}}(\mathcal{D}, \boldsymbol{m}_b, \mathsf{ctx}, \sigma_b)$*.*
4. $\mathcal{A}_2(\pi, \mathsf{st})$ *outputs a bit* b'*.*

We say $\mathcal{A}$ succeeds *if* $b' = b$*, and define* $\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{A},\Pi} \stackrel{\text{def}}{=} |\Pr[\mathcal{A} \text{ succeeds}] - 1/2|$*.*

Anonymity requires that the advantage of any efficient $\mathcal{A}$ should be small. (Since we work in a concrete-security model, we do not include this as part of the definition; instead we give concrete-security bounds when stating our results.) Note that anonymity implies (1) a proof only reveals the set of disclosed attributes, but nothing about non-disclosed attributes; (2) a proof generated by a credential is unlinkable to (issuance of) that credential; and (3) multiple proofs generated using the same credential (whether for the same set of disclosed attributes or not) are unlinkable, beyond whatever linkability is implied by the disclosed credentials themselves. The first condition is evident from the definition, while the second and third are implied by the fact that $\mathcal{A}$ knows σ_0, σ_1 (and so, in particular, can generate proofs using those credentials on its own).

Unforgeability requires that an adversary cannot generate a valid proof for a set of disclosed attributes and context ctx unless (1) it has obtained a credential on a vector matching those attributes, or (2) it has previously seen a valid proof for the same attributes and the same context. In particular, users should not be able to "mix-and-match" attributes from two different credentials. We formalize this by giving an adversary the ability to obtain credentials on attributes of its choice, as well as proofs generated by users with (unknown) credentials on attributes of the adversary's choice.

Definition 3 (Unforgeability). *Consider the following experiment involving a stateful adversary $\mathcal{A}$ and scheme Π:*

1. Setup *is run to generate* params*, and* KeyGen *is run to generate* $(\mathsf{pk}, \mathsf{sk})$*. Both* params *and* pk *are given to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *may repeatedly query the following oracles:*
   - $\mathcal{O}\mathsf{Issue}$*, which on input* $(m_1, \ldots, m_n)$ *returns* $\sigma \leftarrow \mathsf{Issue}_{\mathsf{sk}}(m_1, \ldots, m_n)$*.*
   - $\mathcal{O}\mathsf{Issue}'$*, which on the* i*th query with input* $\boldsymbol{m}_i = (m_{i,j})_{j \in [n]}$ *computes* $\sigma_i \leftarrow \mathsf{Issue}_{\mathsf{sk}}(\boldsymbol{m}_i)$*, but returns nothing to* $\mathcal{A}$*.*
   - $\mathcal{O}\mathsf{Show}$*, which on input* $(i, \mathcal{D}, \mathsf{ctx})$ *returns* $\mathsf{Show}_{\mathsf{pk}}(\mathcal{D}, \boldsymbol{m}_i, \mathsf{ctx}, \sigma_i)$*.*
3. $\mathcal{A}$ *outputs* $\mathcal{D}^* \subseteq [n]$*,* $(m_i^*)_{i \in \mathcal{D}^*}$*,* ctx^**, and* π*.*

$\mathcal{A}$ $\mathsf{succeeds}$ *if (1)* $\mathsf{Vrfy}_{\mathsf{pk}}(\mathcal{D}^*, (m_i^*)_{i \in \mathcal{D}^*}, \mathsf{ctx}^*, \pi) = 1$*, (2)* $\mathcal{A}$ *never made a query* $\mathcal{O}\mathsf{Issue}(\boldsymbol{m}')$ *with* $m'_i = m_i^*$ *for all* $i \in \mathcal{D}^*$*, and (3)* $\mathcal{A}$ *never made a query* $\mathcal{O}\mathsf{Show}(i, \mathcal{D}^*, \mathsf{ctx}^*)$ *where* $m_{i,j} = m_j^*$ *for all* $j \in \mathcal{D}^*$*. Let* $\mathsf{Adv}^{\mathsf{unf}}_{\mathcal{A},\Pi} \stackrel{\text{def}}{=} \Pr[\mathcal{A}\ \mathsf{succeeds}]$.

Note that our definition corresponds to a notion of unforgeability, but could be strengthened to a notion akin to strong unforgeability.

2.3 BBS-Based Anonymous Credentials

We describe BBS-based anonymous credentials. Our description corresponds to the original BBS scheme [5,8,7], recently proven secure [27], rather than the less efficient BBS+ scheme [1]. It consists of the following algorithms:

- Setup: Choose $h_1, \ldots, h_n \leftarrow \mathbb{G}_1$ and return $\boldsymbol{h} \coloneqq (h_1, \ldots, h_n)$.
- KeyGen: Choose $x \leftarrow \mathbb{Z}_p$ and return $(\mathsf{pk}, \mathsf{sk}) \coloneqq (g_2^x, x)$.
- $\mathsf{Issue}_{\mathsf{sk}}(m_1, \ldots, m_n)$: Let $\mathsf{sk} = x \in \mathbb{Z}_p$. Compute $B \coloneqq g_1 \cdot \prod_{i \in [n]} h_i^{m_i}$. Choose $e \leftarrow \mathbb{Z}_p \setminus \{-x\}$ and compute $A \coloneqq B^{1/(x+e)}$. Return $\sigma \coloneqq (A, e)$.
- $\mathsf{VrfyCred}_{\mathsf{pk}}((m_i)_{i \in [n]}, (A, e))$: Compute $B \coloneqq g_1 \cdot \prod_{i \in [n]} h_i^{m_i}$. If $B = 1$, return 0. Otherwise, return $\mathsf{e}(A, \mathsf{pk}) \stackrel{?}{=} \mathsf{e}(A^{-e} \cdot B, g_2)$. (Since we rely on it later, we remark that a valid credential satisfies $A \neq 1$.)
- $\mathsf{Show}_{\mathsf{pk}}(\mathcal{D}, (m_i)_{i \in [n]}, \mathsf{ctx}, \sigma)$: Parse σ as (A, e), and let $B \coloneqq g_1 \cdot \prod_{i \in [n]} h_i^{m_i}$. Choose $r_1, r_2 \leftarrow \mathbb{Z}_p^*$ and set $A' \coloneqq A^{r_1 r_2}$, $\bar{B} \coloneqq B^{r_1}$, and $\bar{A} \coloneqq (A')^{-e} \cdot \bar{B}^{r_2}$. The proof consists of $A', \bar{B}$, and $\bar{A}$, along with a ZKPoK (with tag ctx) that

$$\exists (m_i)_{i \notin \mathcal{D}}, e, r_1, r_2 \ \text{ s.t. } \ \bar{A} = (A')^{-e} \cdot \bar{B}^{r_2} \wedge g_1 \cdot \prod_{i \in \mathcal{D}} h_i^{m_i} = \bar{B}^{1/r_1} \cdot \prod_{i \notin \mathcal{D}} h_i^{-m_i}.$$

  The ZKPoK can be constructed using the techniques in Appendix A.
- $\mathsf{Vrfy}_{\mathsf{pk}}(\mathcal{D}, (m_i)_{i \in \mathcal{D}}, \mathsf{ctx}, \pi)$: Parse π as $A', \bar{B}, \bar{A}$, and a ZKPoK of the statement above. Accept iff (1) $A' \neq 1$, (2) verification of the ZKPoK (with tag ctx) succeeds, and (3) $\mathsf{e}(A', \mathsf{pk}) = \mathsf{e}(\bar{A}, g_2)$.

We remark that Setup does not need to be run by a trusted entity, as $h_1, \ldots, h_n$ can be generated using a random oracle. Also, B could be stored as part of the credential so it does not need to be recomputed each time the credential is used.

The scheme as described does not have perfect correctness since VrfyCred outputs 0 if $g_1 \cdot \prod_{i\in[n]} h_i^{m_i} = 1$; however, for any fixed attributes $\boldsymbol{m} = (m_i)_{i\in[n]}$, the probability (over choice of $\boldsymbol{h}$ output by Setup) that this occurs is $1/p$. In fact, even if we allow an adversary to choose the attributes after observing the public parameters, finding $\boldsymbol{m}$ such that $g_1 \cdot \prod_{i\in[n]} h_i^{m_i} = 1$ is as hard as solving the discrete-logarithm problem in $\mathbb{G}_1$.

3 Issuer-Hiding Anonymous Credentials

We extend the definitions in the previous section to incorporate issuer hiding. Although our syntax differs slightly, our definition of anonymity is equivalent to the definition of anonymity by Sanders and Traoré [24]. Our definition of unforgeability is stronger than theirs, however, since we additionally allow the adversary to obtain proofs generated by users with credentials unknown to the adversary, as in other prior work [3].

We begin by introducing the additional algorithms needed in our setting. An issuer-hiding scheme should be compatible with some underlying anonymous-credential scheme, so we keep algorithms KeyGen, Issue, and VrfyCred (cf. Definition 1) unchanged. Setup may need to be modified if additional setup is needed to support issuer hiding (though any such additional setup will not be used by KeyGen, Issue, or VrfyCred). The Show and Vrfy algorithms can continue to be used as before if a user is willing to reveal the issuer of their credential; however, we now overload those algorithms to support issuer hiding, as we describe next.

As in prior work [24], a verifier who is willing to accept credentials issued by any issuer in a set $\mathcal{P}$ with $|\mathcal{P}| > 1$ (where issuers are identified with their public keys) runs $\mathsf{SetPolicy}(\mathcal{P})$ to generate a (public) policy key $\mathsf{pk}_{\mathcal{P}}$ along with an associated (private) verification key $\mathsf{sk}_{\mathcal{P}}$. A user can audit a policy key by running AuditPolicy. A user holding a credential σ on attributes $(m_1, \ldots, m_n)$, where σ was issued by an issuer whose public key is in $\mathcal{P}$, can use $\mathsf{Show}_{\mathsf{pk}_{\mathcal{P}}}$ to selectively disclose some subset $(m_i)_{i\in\mathcal{D}}$ of those attributes to that verifier; the resulting proof can be verified using $\mathsf{sk}_{\mathcal{P}}$.

Definition 4 (Issuer-Hiding Anonymous Credentials). *An issuer-hiding anonymous-credential scheme consists of algorithms* Setup, KeyGen, Issue, *and* VrfyCred *as in Definition 1 with the following additional/extended algorithms:*

- SetPolicy *takes as input a set $\mathcal{P}$ of issuers' public keys, and outputs a policy key $\mathsf{pk}_{\mathcal{P}}$ and associated verification key $\mathsf{sk}_{\mathcal{P}}$.*
- AuditPolicy *takes as input a set $\mathcal{P}$ of issuers' public keys and a policy key $\mathsf{pk}_{\mathcal{P}}$ and outputs a bit.*
- Show *now takes as input a policy key $\mathsf{pk}_{\mathcal{P}}$, a set of issuers' public keys $\mathcal{P}$, a set $\mathcal{D} \subseteq [n]$, attributes $m_1, \ldots, m_n$, context* ctx*, a credential σ, and a public key $\mathsf{pk} \in \mathcal{P}$. It outputs a proof π.*
- Vrfy *now takes as input a verification key $\mathsf{sk}_{\mathcal{P}}$, a set of public keys $\mathcal{P}$, a set $\mathcal{D} \subseteq [n]$, attributes $(m_i)_{i\in\mathcal{D}}$, context* ctx*, and a proof π. It outputs a bit.*

We assume $|\mathcal{P}| > 1$ in all the above.[3] *In addition to the correctness conditions from Definition 1, we additionally require:*

1. *For all* params *output by* Setup*, any keys* $\{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i\in[\ell]}$ *output by* KeyGen*, and any* $(\mathsf{pk}_\mathcal{P}, \mathsf{sk}_\mathcal{P})$ *output by* $\mathsf{SetPolicy}(\mathcal{P})$ *(where* $\mathcal{P} = \{\mathsf{pk}_i\}_{i\in[\ell]}$*) we have* $\mathsf{AuditPolicy}(\mathcal{P}, \mathsf{pk}_\mathcal{P}) = 1$*.*
2. *For all* $\boldsymbol{m} = (m_i)_{i\in[n]}$*, all* params *output by* Setup*, any* $\{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i\in[\ell]}$ *output by* KeyGen*, any* $(\mathsf{pk}_\mathcal{P}, \mathsf{sk}_\mathcal{P})$ *output by* $\mathsf{SetPolicy}(\mathcal{P})$ *(where* $\mathcal{P} = \{\mathsf{pk}_i\}_{i\in[\ell]}$*) any* $\mathcal{D} \subseteq [n]$*, any* ctx*, any* $j \in [\ell]$*, any* σ *with* $\mathsf{VrfyCred}_{\mathsf{pk}_j}(\boldsymbol{m}, \sigma) = 1$*, and all* π *output by* $\mathsf{Show}_{\mathsf{pk}_\mathcal{P}}(\mathcal{P}, \mathcal{D}, \boldsymbol{m}, \mathsf{ctx}, \sigma, \mathsf{pk}_j)$*, we have*

$$\mathsf{Vrfy}_{\mathsf{sk}_\mathcal{P}}(\mathcal{P}, \mathcal{D}, (m_i)_{i\in\mathcal{D}}, \mathsf{ctx}, \pi) = 1.$$

We define issuer hiding by strengthening the definition of anonymity (Definition 2): in addition to allowing an adversary to specify two sets of attributes and two credentials, we now also allow the adversary to specify two public keys.

Definition 5 (Issuer-Hiding Anonymity). *Consider the following experiment involving an adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *and scheme* Π*:*

1. Setup *is run to generate* params*.*
2. $\mathcal{A}_1(\mathsf{params})$ *outputs* $(\mathcal{P}, \mathsf{pk}_\mathcal{P}, \mathcal{D}, \mathsf{ctx}, \boldsymbol{m}_0, \sigma_0, \mathsf{pk}_0, \boldsymbol{m}_1, \sigma_1, \mathsf{pk}_1)$ *and* st *where:*
   - $\mathsf{pk}_0, \mathsf{pk}_1 \in \mathcal{P}$*,*
   - $\mathsf{AuditPolicy}(\mathcal{P}, \mathsf{pk}_\mathcal{P}) = 1$*,*
   - $\mathsf{VrfyCred}_{\mathsf{pk}_0}(\boldsymbol{m}_0, \sigma_0) = \mathsf{VrfyCred}_{\mathsf{pk}_1}(\boldsymbol{m}_1, \sigma_1) = 1$*, and*
   - $m_{0,i} = m_{1,i}$ *for all* $i \in \mathcal{D}$*.*
3. *Choose* $b \leftarrow \{0, 1\}$ *and compute* $\pi \leftarrow \mathsf{Show}_{\mathsf{pk}_\mathcal{P}}(\mathcal{P}, \mathcal{D}, \boldsymbol{m}_b, \mathsf{ctx}, \sigma_b, \mathsf{pk}_b)$*.*
4. $\mathcal{A}_2(\pi, \mathsf{st})$ *outputs a bit* b'*.*

We say $\mathcal{A}$ succeeds *if* $b' = b$*, and define* $\mathsf{Adv}^{\mathsf{ih\text{-}anon}}_{\mathcal{A},\Pi} \stackrel{\text{def}}{=} |\Pr[\mathcal{A} \text{ succeeds}] - 1/2|$*.*

Note that Definition 5 implies Definition 2 (assuming the case of $|\mathcal{P}| = 1$ is handled as discussed in footnote 3).

For unforgeability we modify Definition 3 in the natural way.

Definition 6 (Unforgeability). *Consider the following experiment involving a stateful adversary* $\mathcal{A}$ *and scheme* Π*:*

1. Setup *is run to generate* params*, which are given to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *may repeatedly query the following oracles:*
   - $\mathcal{O}$KeyGen*, which on the* i*th such query computes* $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow$ KeyGen *and returns* pk_i *to* $\mathcal{A}$*.*
   - $\mathcal{O}$Issue*, which on input* $(j, m_1, \ldots, m_n)$ *returns* $\mathsf{Issue}_{\mathsf{sk}_j}(m_1, \ldots, m_n)$*.*
   - $\mathcal{O}\mathsf{Issue}'$*, which on the* i*th such query* $(j, \boldsymbol{m}_i)$ *computes* $\sigma_i \leftarrow \mathsf{Issue}_{\mathsf{sk}_j}(\boldsymbol{m}_i)$*, but returns nothing to* $\mathcal{A}$*. (Here,* sk_j *is the secret key generated in the* j*th query to* $\mathcal{O}$KeyGen*.) We say* σ_i *is* associated with pk_j *in this case.*

[3] When $|\mathcal{P}| = 1$ then SetPolicy and AuditPolicy are trivial algorithms, and Show and Vrfy from the underlying scheme can be used.

– $\mathcal{O}\mathsf{Show}$, *which on input* $(i, \mathcal{P}, \mathsf{pk}_{\mathcal{P}}, \mathcal{D}, \mathsf{ctx})$ *returns* $\perp$ *if* σ_i *is associated with a public key* $\mathsf{pk}_j \notin \mathcal{P}$ *or* $\mathsf{AuditPolicy}(\mathcal{P}, \mathsf{pk}_{\mathcal{P}}) = 0$*, and otherwise returns* $\mathsf{Show}_{\mathsf{pk}_{\mathcal{P}}}(\mathcal{P}, \mathcal{D}, \boldsymbol{m}_i, \mathsf{ctx}, \sigma_i, \mathsf{pk}_j)$*. (Here,* σ_i *is the credential generated in the* i*th query to* $\mathcal{O}\mathsf{Issue}'$*.)*
– $\mathcal{O}\mathsf{SetPolicy}$, *which given* $\mathcal{P}^* \subseteq \{\mathsf{pk}_i\}$ *runs* $(\mathsf{pk}_{\mathcal{P}^*}, \mathsf{sk}_{\mathcal{P}^*}) \leftarrow \mathsf{SetPolicy}(\mathcal{P}^*)$ *and returns* $\mathsf{pk}_{\mathcal{P}^*}$*. This oracle may be called only once.*
– $\mathcal{O}\mathsf{Vrfy}$, *which on input* $(\mathcal{D}, (m_i)_{i\in\mathcal{D}}, \mathsf{ctx}, \pi)$ *returns*

$$\mathsf{Vrfy}_{\mathsf{sk}_{\mathcal{P}^*}}(\mathcal{P}^*, \mathcal{D}, (m_i)_{i\in\mathcal{D}}, \mathsf{ctx}, \pi),$$

assuming query $\mathcal{O}\mathsf{SetPolicy}(\mathcal{P}^*)$ *was already made to define* $\mathcal{P}^*, \mathsf{sk}_{\mathcal{P}^*}$*.*

3. $\mathcal{A}$ *outputs* $\mathcal{D}^*$*,* $(m_i^*)_{i\in\mathcal{D}^*}$*,* ctx^**, and* π*.*

Let $\mathcal{P}^*, \mathsf{pk}_{\mathcal{P}^*}, \mathsf{sk}_{\mathcal{P}^*}$ *be as defined by the* $\mathcal{O}\mathsf{SetPolicy}$ *query. We say* $\mathcal{A}$ succeeds *if*

(1) $\mathsf{Vrfy}_{\mathsf{sk}_{\mathcal{P}^*}}(\mathcal{P}^*, \mathcal{D}^*, (m_i^*)_{i\in\mathcal{D}^*}, \mathsf{ctx}^*, \pi) = 1$*,*
(2) $\mathcal{A}$ *never made a query* $\mathcal{O}\mathsf{Issue}(j, m_1', \ldots, m_n')$ *with* $\mathsf{pk}_j \in \mathcal{P}^*$ *and* $m_i' = m_i^*$ *for all* $i \in \mathcal{D}^*$*,*
(3) $\mathcal{A}$ *never made a query* $\mathcal{O}\mathsf{Show}(i, \mathcal{P}^*, \mathsf{pk}_{\mathcal{P}^*}, \mathcal{D}^*, \mathsf{ctx}^*)$ *where* σ_i *is associated with a public key in* $\mathcal{P}^*$*, and* $m_{i,j} = m_j^*$ *for all* $j \in \mathcal{D}^*$*.*

Let $\mathsf{Adv}^{\mathsf{ih\text{-}unf}}_{\mathcal{A},\Pi} \stackrel{\text{def}}{=} \Pr[\mathcal{A} \text{ succeeds}]$.

Definition 6 implies Definition 3 assuming the case of $|\mathcal{P}| = 1$ is handled as discussed in footnote 3.

4 Issuer-Hiding BBS-Based Anonymous Credentials

We now describe how to incorporate issuer hiding into BBS-based anonymous credentials. Setup, KeyGen, Issue, and VrfyCred are unchanged from the underlying scheme (cf. Section 2.3). Other algorithms are defined as follows.

$\mathsf{SetPolicy}(\mathcal{P})$: Parse $\mathcal{P}$ as $\{\mathsf{pk}_i\}_{i\in[\ell]}$. Choose $a \leftarrow \mathbb{Z}_p^*$ and $b \leftarrow \mathbb{Z}_p$, and then compute $S := g_2^a$. For $i \in [\ell]$, set $T_i := \left(\mathsf{pk}_i \cdot g_2^b\right)^a$. Set $\mathsf{sk}_{\mathcal{P}} := (a, b)$. The policy key $\mathsf{pk}_{\mathcal{P}}$ includes $(S, \{T_i\}_{i\in[\ell]})$ plus a ZKPoK that

$$\exists a, b \text{ s.t. } S = g_2^a \ \wedge\ \forall i : T_i = (\mathsf{pk}_i \cdot g_2^b)^a. \tag{1}$$

The ZKPoK can be constructed using the techniques in Appendix A.

$\mathsf{AuditPolicy}(\mathcal{P}, \mathsf{pk}_{\mathcal{P}})$: Parse $\mathsf{pk}_{\mathcal{P}}$ as $(S, \{T_i\}_{i\in[\ell]})$ plus a ZKPoK of statement (1). Accept if $\ell = |\mathcal{P}|$, $S \neq 1$, and verification of the ZKPoK succeeds.

$\mathsf{Show}_{\mathsf{pk}_{\mathcal{P}}}(\mathcal{P}, \mathcal{D}, (m_i)_{i\in[n]}, \mathsf{ctx}, \sigma, \mathsf{pk})$: If $|\mathcal{P}| = 1$, the original Show algorithm is used, so we assume $|\mathcal{P}| > 1$. Parse $\mathcal{P}$ as $\{\mathsf{pk}_i\}_{i\in[\ell]}$ and σ as (A, e), let i^* be s.t. $\mathsf{pk} = \mathsf{pk}_{i^*}$, and let $(S, \{T_i\}_{i\in[\ell]})$ be the elements included in $\mathsf{pk}_{\mathcal{P}}$. Choose $r_1, r_2 \leftarrow \mathbb{Z}_p^*$ and $t \leftarrow \mathbb{Z}_p$. Compute $B := g_1 \prod_{i\in[n]} h_i^{m_i}$, $A' := A^{r_1 r_2}$, $\bar{B} := B^{r_1}$,

$\bar{A} := (A')^{-e-t}\bar{B}^{r_2}$, and $\tilde{\sigma} := S^t \cdot \prod_{i \neq i^*} T_i$. The proof consists of $A', \bar{B}, \bar{A}$, and $\tilde{\sigma}$, along with a ZKPoK (with tag $(\mathcal{P}, \mathsf{pk}_{\mathcal{P}}, \mathsf{ctx})$) that

$$\begin{aligned} &\exists (m_i)_{i \notin \mathcal{D}}, \tilde{e}, r_1, r_2 \text{ s.t.} \\ &\bar{A} = (A')^{-\tilde{e}} \cdot \bar{B}^{r_2} \ \wedge \ g_1 \cdot \prod_{i \in \mathcal{D}} h_i^{m_i} = \bar{B}^{1/r_1} \cdot \prod_{i \notin \mathcal{D}} h_i^{-m_i}. \end{aligned} \tag{2}$$

$\mathsf{Vrfy}_{\mathsf{sk}_{\mathcal{P}}}(\mathcal{P}, \mathcal{D}, (m_i)_{i \in \mathcal{D}}, \mathsf{ctx}, \pi)$: If $|\mathcal{P}| = 1$, the original Vrfy algorithm is used, so we assume $|\mathcal{P}| = \ell > 1$. Parse π as $A', \bar{B}, \bar{A}, \tilde{\sigma}$, and a ZKPoK of statement (2), and parse $\mathsf{sk}_{\mathcal{P}}$ as (a, b). Accept iff (1) $A' \neq 1$, (2) verification of the ZKPoK (with tag $(\mathcal{P}, \mathsf{pk}_{\mathcal{P}}, \mathsf{ctx})$) succeeds, and (3) it holds that

$$\mathsf{e}\left(A', \ \tilde{\sigma}^{-1/a} \cdot g_2^{(\ell-1)\cdot b} \cdot \textstyle\prod_{i \in [\ell]} \mathsf{pk}_i\right) \stackrel{?}{=} \mathsf{e}(\bar{A}, g_2).$$

Note that the verifier can store $g_2^{(\ell-1)\cdot b} \cdot \prod_{i \in [\ell]} \mathsf{pk}_i$ as part of $\mathsf{sk}_{\mathcal{P}}$ to avoid having to compute it each time. In fact, if it stores $R = \left(g_2^{(\ell-1)\cdot b} \cdot \prod_{i \in [\ell]} \mathsf{pk}_i\right)^{-a}$ then the third step of verification can be rewritten as

$$\mathsf{e}\left((A')^{-1/a}, \ \tilde{\sigma} \cdot R\right) \stackrel{?}{=} \mathsf{e}(\bar{A}, g_2),$$

and the verifier can avoid computing any exponentiations in $\mathbb{G}_2$.

Correctness. It is immediate that the first correctness condition from Definition 4 holds. To verify the second condition, fix $\boldsymbol{h} = (h_i)_{i \in [n]}$, $\mathcal{P} = \{\mathsf{pk}_i\}_{i \in [\ell]}$, keys $(\mathsf{pk}_{\mathcal{P}}, \mathsf{sk}_{\mathcal{P}})$ output by $\mathsf{SetPolicy}(\mathcal{P})$, a credential σ on attributes $\boldsymbol{m} = (m_i)_{i \in [n]}$ for which $\mathsf{VrfyCred}_{\mathsf{pk}_{i^*}}(\boldsymbol{m}, \sigma) = 1$ for some $i^* \in [\ell]$, a disclosure set $\mathcal{D}$, and ctx. Consider the proof output by $\mathsf{Show}_{\mathsf{pk}_{\mathcal{P}}}(\mathcal{P}, \mathcal{D}, \boldsymbol{m}, \mathsf{ctx}, \sigma, \mathsf{pk}_{i^*})$ that includes $A', \bar{B}, \bar{A}$, and $\tilde{\sigma}$. Verification of the ZKPoK clearly succeeds. Since $\mathsf{VrfyCred}_{\mathsf{pk}_{i^*}}(\boldsymbol{m}, \sigma) = 1$, we know that $A \neq 1$. It follows that $A' \neq 1$. Finally, we have

$$\begin{aligned} &\mathsf{e}\left(A', \ \tilde{\sigma}^{-1/a} \cdot g_2^{(\ell-1)\cdot b} \cdot \textstyle\prod_{i \in [\ell]} \mathsf{pk}_i\right) \\ &= \mathsf{e}\left(A', \ \left(S^t \cdot \textstyle\prod_{i \neq i^*} T_i\right)^{-1/a} \cdot g_2^{(\ell-1)\cdot b} \cdot \textstyle\prod_{i \in [\ell]} \mathsf{pk}_i\right) \\ &= \mathsf{e}\left(A', \ \left(g_2^{at} \cdot \textstyle\prod_{i \neq i^*} \left(\mathsf{pk}_i \cdot g_2^b\right)^a\right)^{-1/a} \cdot g_2^{(\ell-1)\cdot b} \cdot \textstyle\prod_{i \in [\ell]} \mathsf{pk}_i\right) \\ &= \mathsf{e}\left(A', \ g_2^{-t} \cdot \textstyle\prod_{i \neq i^*} \left(\mathsf{pk}_i \cdot g_2^b\right)^{-1} \cdot g_2^{(\ell-1)\cdot b} \cdot \textstyle\prod_{i \in [\ell]} \mathsf{pk}_i\right) \\ &= \mathsf{e}\left(A', \ g_2^{-t} \cdot \textstyle\prod_{i \neq i^*} \mathsf{pk}_i^{-1} \cdot g_2^{-(\ell-1)\cdot b} \cdot g_2^{(\ell-1)\cdot b} \cdot \textstyle\prod_{i \in [\ell]} \mathsf{pk}_i\right) \\ &= \mathsf{e}\left(A', \ g_2^{-t} \cdot \mathsf{pk}_{i^*}\right) \ = \ \mathsf{e}\left(A', g_2^{-t}\right) \cdot \mathsf{e}\left(A', \mathsf{pk}_{i^*}\right). \end{aligned}$$

As in the underlying BBS scheme, $\mathsf{e}(A', \mathsf{pk}_{i^*}) = \mathsf{e}((A')^{-e}\bar{B}^{r_2}, g_2)$. Thus,

$$\begin{aligned} \mathsf{e}\left(A', g_2^{-t}\right) \cdot \mathsf{e}\left(A', \mathsf{pk}_{i^*}\right) &= \mathsf{e}\left((A')^{-t}, g_2\right) \cdot \mathsf{e}((A')^{-e}\bar{B}^{r_2}, g_2) \\ &= \mathsf{e}\left((A')^{-e-t}\bar{B}^{r_2}, g_2\right) \ = \ \mathsf{e}(\bar{A}, g_2), \end{aligned}$$

and verification succeeds.

Simulatability. It is worth noting that, just as in the case of standard BBS-based anonymous credentials, the proofs output by the Show algorithm are simulatable given some auxiliary information. Specifically, consider the output of $\mathsf{Show}_{\mathsf{pk}_\mathcal{P}}(\mathcal{P}, \mathcal{D}, \boldsymbol{m}, \mathsf{ctx}, \sigma, \mathsf{pk})$ where $\mathcal{P} = \{\mathsf{pk}_i\}$, index i^* is such that $\mathsf{pk} = \mathsf{pk}_{i^*}$, and $(S, \{T_i\})$ are the elements included in $\mathsf{pk}_\mathcal{P}$. Assume further that auxiliary information $\mathsf{pk}' = g_1^x$ (where $\mathsf{pk} = g_2^x$) is available. The ZKPoK can clearly be simulated, and $A', \bar{B}, \bar{A}$, and $\tilde{\sigma}$ can be perfectly simulated by choosing $\bar{B} \leftarrow \mathbb{G}_1^*$, $r \leftarrow \mathbb{Z}_p^*$, and $t \leftarrow \mathbb{Z}_p$, and outputting

$$A' \coloneqq g_1^r, \quad \bar{B}, \quad \bar{A} \coloneqq (\mathsf{pk}')^r \cdot (A')^{-t}, \quad \tilde{\sigma} \coloneqq S^t \cdot \prod_{i \neq i^*} T_i.$$

In fact, for the above it suffices to be given (A, A^x) for any $A \in \mathbb{G}_1^*$, and such a pair can be computed from a credential $(A = B^{1/(x+e)}, e)$ for *any* set of attributes by noting that $A^x = B \cdot A^{-e}$.

5 Security Proofs

Here we prove issuer-hiding anonymity and unforgeability of the scheme from Section 4. Throughout, we let Π denote that scheme.

5.1 Issuer-Hiding Anonymity

We show that our scheme satisfies what we call "everlasting" issuer-hiding anonymity. Specifically, so long as $\mathcal{A}_1$ cannot violate soundness of the ZKPoK when generating the policy key $\mathsf{pk}_\mathcal{P}$, then even an unbounded $\mathcal{A}_2$ cannot violate issuer-hiding anonymity for proofs based on that policy key (i.e., issuer-hiding anonymity then holds *unconditionally*). This has two important consequences:

- If the ZKPoK satisfies soundness against classical adversaries and large-scale quantum computers are first built at some time T, then issuer-hiding anonymity continues to hold (even against post-quantum adversaries) for any user-generated proofs produced before time T.
- If a resource-bounded (possibly quantum) adversary cannot violate soundness of the ZKPoK in some initial period of time, then user-generated proofs produced during that time retain issuer-hiding anonymity forever.

In particular, the above rule out "store-now/deanonymize-later" attacks that may be a concern in practice.

Theorem 7. *Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary. Then either $\mathcal{A}_1$ violates soundness of the ZKPoK for statement (1), or $\mathsf{Adv}^{\mathsf{ih\text{-}anon}}_{\mathcal{A},\Pi} = 0$.*

Proof. Fix an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in the experiment from Definition 5, let $\boldsymbol{h} = (h_i)_{i \in [n]}$ be the output of Setup, and let $(\mathcal{P}, \mathsf{pk}_\mathcal{P}, \mathcal{D}, \mathsf{ctx}, \boldsymbol{m}_0, \sigma_0, \mathsf{pk}_0, \boldsymbol{m}_1, \sigma_1, \mathsf{pk}_1)$ and st be the output of $\mathcal{A}_1(\boldsymbol{h})$. Assume $\mathcal{A}_1$ does not violate soundness of the ZKPoK. Since $\mathsf{AuditPolicy}(\mathcal{P}, \mathsf{pk}_\mathcal{P}) = 1$, this means that if $\mathcal{P} = \{\mathsf{pk}'_i\}_{i \in [\ell]}$ and $\mathsf{pk}_\mathcal{P}$

includes $(S, \{T_i\}_{i\in[\ell]})$, then $S \neq 1$ and there exist a, b' such that $S = g_2^a$ and $T_i = (\mathsf{pk}'_i \cdot g_2^{b'})^a$ for all i. We show that the outputs of $\mathsf{Show}_{\mathsf{pk}_\mathcal{P}}(\mathcal{P}, \mathcal{D}, \boldsymbol{m}_0, \mathsf{ctx}, \sigma_0, \mathsf{pk}_0)$ and $\mathsf{Show}_{\mathsf{pk}_\mathcal{P}}(\mathcal{P}, \mathcal{D}, \boldsymbol{m}_1, \mathsf{ctx}, \sigma_1, \mathsf{pk}_1)$ are identically distributed. To show this, it suffices to show that the distributions of $A', \bar{B}, \bar{A}, \tilde{\sigma}$ in each case are identical, since we assume the ZKPoK used by Show satisfies perfect zero knowledge.

Let $\sigma_b = (A_b, e_b)$ and $B_b = g_1 \cdot \prod_{i\in[n]} h_i^{m_{b,i}}$ for $b \in \{0, 1\}$. Since

$$\mathsf{VrfyCred}_{\mathsf{pk}_0}(\boldsymbol{m}_0, \sigma_0) = \mathsf{VrfyCred}_{\mathsf{pk}_1}(\boldsymbol{m}_1, \sigma_1) = 1,$$

we know that for $b \in \{0, 1\}$ we have $A_b, B_b \neq 1$. Then:

- Elements $A', \bar{B}$ output by Show are independent and uniform in $\mathbb{G}_1 \setminus \{1\}$ for either b since $A_b, B_b \neq 1$ and r_1, r_2 are independent and uniform in $\mathbb{Z}_p^*$.
- Element $\tilde{\sigma}$ is uniform in $\mathbb{G}_2$, even conditioned on $A', \bar{B}$, for either choice of b. This is because $S \neq 1$ and t is uniform in $\mathbb{Z}_p$ and independent of r_1, r_2.
- For a given $A', \bar{B}$, and $\tilde{\sigma}$, there is a unique $\bar{A}$ for which verification succeeds (and, by correctness, that $\bar{A}$ is always output by Show for either choice of b). This follows immediately from the pairing check

  $$\mathsf{e}\left(A',\ \tilde{\sigma}^{-1/a} \cdot g_2^{(\ell-1)b'} \cdot \textstyle\prod_{i\in[\ell]} \mathsf{pk}'_i\right) = \mathsf{e}(\bar{A}, g_2)$$

  done as part of verification. Note that here we rely on the fact that a, b are determined by $\mathsf{pk}_\mathcal{P}$ (which in turn relies on the assumption that $\mathcal{A}_1$ did not violate soundness of the ZKPoK).

This concludes the proof. □

In Appendix B we show that soundness of the ZKPoK is essential; specifically, if the ZKPoK is omitted then there is an attack on issuer-hiding anonymity.

5.2 Unforgeability

We show that our scheme is (unconditionally) unforgeable in the generic group model (GGM) for pairing-based groups, assuming the ZKPoK used to prove statement (2) is simulation extractable.

Due to the structure of BBS signatures, when using the standard GGM technique of replacing secret exponents of group elements with indeterminates, we obtain rational functions rather than polynomials. To handle these, we need a generalization of the Schwartz–Zippel lemma to rational functions. In the proofs that follow, we use "≡" for identity of rational functions, and "=" for equality of values in $\mathbb{Z}_p$.

Lemma 8. *Let $f = \frac{P}{Q}, f' = \frac{P'}{Q'} \in \mathbb{Z}_p(\mathsf{X}_1, \ldots, \mathsf{X}_n)$ be rational functions, where $P, P', Q, Q' \in \mathbb{Z}_p[\mathsf{X}_1, \ldots, \mathsf{X}_n]$ are polynomials of total degree $d_P, d_{P'}, d_Q, d_{Q'}$, respectively, $Q, Q' \not\equiv 0$, and $PQ' - P'Q \not\equiv 0$. Let $\mathcal{X} \stackrel{\text{def}}{=} \{\boldsymbol{x} \in \mathbb{Z}_p^n \mid Q(\boldsymbol{x}) \neq 0,$*

$Q'(\boldsymbol{x}) \neq 0\}$ *be the set of points on which both* Q *and* Q' *are nonzero, and let* $\mathcal{S} \subseteq \mathcal{X}$ *be non-empty. Then*

$$\Pr\big[\boldsymbol{x} \leftarrow \mathcal{S} : f(\boldsymbol{x}) = f'(\boldsymbol{x})\big] \leq \frac{\max\{d_P + d_{Q'}, d_{P'} + d_Q\} \cdot p^{n-1}}{|\mathcal{S}|}.$$

Proof. When $\frac{P(\boldsymbol{x})}{Q(\boldsymbol{x})} = \frac{P'(\boldsymbol{x})}{Q'(\boldsymbol{x})}$ for $x \in \mathcal{S} \subseteq \mathcal{X}$, then $Q(\boldsymbol{x}), Q'(\boldsymbol{x}) \neq 0$ and

$$P(\boldsymbol{x})Q'(\boldsymbol{x}) - P'(\boldsymbol{x})Q(\boldsymbol{x}) = 0.$$

Let $\mathcal{Z} \stackrel{\text{def}}{=} \{\boldsymbol{x} \in \mathbb{Z}_p^n \mid P(\boldsymbol{x})Q'(\boldsymbol{x}) - P'(\boldsymbol{x})Q(\boldsymbol{x}) = 0\}$ be the set of roots of the nonzero polynomial $PQ' - P'Q$. Since $PQ' - P'Q$ has total degree at most

$$\max\{d_P + d_{Q'}, d_{P'} + d_Q\},$$

the Schwartz–Zippel lemma implies $|\mathcal{Z}| \leq \max\{d_P + d_{Q'}, d_{P'} + d_Q\} \cdot p^{n-1}$. The probability that $f(\boldsymbol{x}) = f'(\boldsymbol{x})$ is at most $|\mathcal{Z}|/|\mathcal{S}|$, giving the claimed bound. □

Theorem 9. *Model* $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ *as generic pairing-based groups, and let* $\mathcal{A}$ *be an adversary making at most* $q_{\mathbb{G}}$ *group-operation queries,* q_K *key-generation queries,* q_I *issuance queries,* q_S *show queries, and* q_V *verification queries with* $q_I + 2q_S + 1 < p$*, and querying* $\mathcal{O}\mathsf{SetPolicy}$ *on a set* $\mathcal{P}$ *of* ℓ *public keys. Assume the ZKPoKs for statements (1) and (2) have perfect zero knowledge. Unless* $\mathcal{A}$ *violates simulation extractability of the ZKPoK for statement (2), we have*

$$\mathsf{Adv}^{\mathsf{ih\text{-}unf}}_{\mathcal{A},\Pi} \leq \frac{(2q_I + 2q_S + 5)(q_{\mathbb{G}} + q_K + q_I + 4q_S + q_V + \ell + n + 7)^2}{2(p - q_I - 2q_S - 1)} + \frac{q_I^2 - q_I}{2(p-1)}.$$

Proof. Fix an adversary $\mathcal{A}$ in the experiment from Definition 6, let $(h_i)_{i\in[n]}$ be the output of Setup, and let $(\mathcal{D}, (m_i)_{i\in\mathcal{D}}, \mathsf{ctx}, \pi)$ be the output of $\mathcal{A}$. Let $(\mathsf{pk}_i)_{i\in[q_K]}$ be the public keys output by $\mathcal{O}\mathsf{KeyGen}$ in response to $\mathcal{A}$'s queries and, for $j \in [q_I]$, let $(m_{j,i})_{i\in[n]}$ be the attributes on which $\mathcal{A}$ queried $\mathcal{O}\mathsf{Issue}$, resulting in credential (A_j, e_j). For the set of ℓ public keys $\mathcal{P}$ on which $\mathcal{A}$ queries $\mathcal{O}\mathsf{SetPolicy}$, let $I_{\mathcal{P}} \stackrel{\text{def}}{=} \{i \mid \mathsf{pk}_i \in \mathcal{P}\} \subseteq [q_K]$ be the index set for those keys.

We argue that in the GGM, if $\mathsf{Vrfy}_{\mathsf{sk}_{\mathcal{P}}}(\mathcal{P}, \mathcal{D}, (m_i)_{i\in\mathcal{D}}, \mathsf{ctx}, \pi) = 1$ and $\mathcal{A}$ never made a query $\mathcal{O}\mathsf{Show}(\cdot, \mathcal{P}, \mathsf{pk}_{\mathcal{P}}, \mathcal{D}, \mathsf{ctx})$ with associated query $\mathcal{O}\mathsf{Issue}'(i', \boldsymbol{m}')$ for which $\mathsf{pk}_{i'} \in \mathcal{P}$ and $m'_i = m_i$ for all $i \in \mathcal{D}$, then (with high probability, and assuming $\mathcal{A}$ does not violate simulation extractability) there exists $i^* \in I_{\mathcal{P}}$ such that the adversary queried $\mathcal{O}\mathsf{Issue}$ on input $(i^*, \boldsymbol{m}^*)$ with $m^*_i = m_i$ for all $i \in \mathcal{D}$, i.e., $\mathcal{A}$ obtained a valid credential with respect to $\mathsf{pk}_{i^*} \in \mathcal{P}$ on the attributes it disclosed. Thus, the adversary does not succeed in the experiment from Definition 6.

We consider a sequence of experiments Expt_0, Expt_1 (given in Figure 1), and Expt_2 (given in Figure 2), and let $\Pr[\mathsf{Expt}_i]$ be the probability that Expt_i returns 1. We model the GGM using functions $\{\mathsf{Label}_i\}_{i\in\{1,2,T\}}$ that assign random labels to elements in the three different groups. Formally, we define $\mathsf{Label}_i(x)$ for $x \in \mathbb{Z}_p$ as follows (where σ_i is initialized to $\emptyset$ at the beginning of the experiment):

If $\sigma_i(x)$ is already defined, return $y := \sigma_i(x)$. Otherwise, choose a random, unused label $y \leftarrow \{0,1\}^{2\lceil \log p \rceil} \setminus \mathrm{range}(\sigma_i)$, set $\sigma_i(x) := y$, and return y.

To simplify notation, we use $[\cdot]_i$ as shorthand for $\sigma_i^{-1}(\cdot)$. $\mathcal{A}$ can generically carry out group operations using oracles $\{\mathcal{O}\mathsf{Group}_i\}_{i\in\{1,2,T\}}$ and $\mathcal{O}\mathsf{Pair}$ defined as follows: $\mathcal{O}\mathsf{Group}_i(y_1, y_2)$ returns $y_3 \leftarrow \mathsf{Label}_i([y_1]_i + [y_2]_i)$ and $\mathcal{O}\mathsf{Pair}(y_1, y_2)$ returns $y_T \leftarrow \mathsf{Label}_T([y_1]_1 \cdot [y_2]_2)$. (We assume for simplicity that the adversary only makes queries using group labels it previously received. Since labels are sampled from $\{0,1\}^{2\lceil \log p \rceil}$, the probability that the adversary can generate a "new" valid label is negligible.)

We claim that if $\mathcal{A}$ does not violate simulation extractability of the ZKPoK for statement (2), then

$$\mathsf{Adv}^{\mathsf{ih\text{-}unf}}_{\mathcal{A},\Pi} = \Pr[\mathsf{Expt}_0]. \tag{3}$$

To see this, note that Expt_0 corresponds to the original unforgeability experiment from Definition 6 in the GGM, except that it returns 0 in line 8 if extraction of the witness for statement (2) fails. However, if the check in line 5 succeeds then verification of the ZKPoK output by $\mathcal{A}$ succeeds, and thus (assuming $\mathcal{A}$ does not violate simulation extractability) extraction succeeds unless the ZKPoK output by $\mathcal{A}$ is not for a "fresh" statement-tag pair. But the only way the latter can occur is if $\mathcal{A}$ previously queried $\mathcal{O}\mathsf{Show}(j, \mathcal{P}, \mathsf{pk}_{\mathcal{P}}, \mathcal{D}, \mathsf{ctx})$ with $\mathsf{pk}_{i'_j} \in \mathcal{P}$ and $m'_{j,i} = m_i$ for all $i \in \mathcal{D}$ (note that $\mathcal{D}$ and $(m_i)_{i\in\mathcal{D}}$ are part of the statement being proved), in which case the check in line 7 would already have failed.

We now analyze the differences between successive experiments.

$\mathsf{Expt}_0 \to \mathsf{Expt}_1$. The only difference between the first two experiments is that Expt_1 aborts in line 1 of $\mathcal{O}\mathsf{Issue}$ if there is a collision among the $(e_j)_{j\in[q_I]}$. This occurs with probability at most $\frac{q_I(q_I-1)}{2(p-1)}$, since each e_j is sampled from a set of size $p-1$. Thus,

$$\Pr[\mathsf{Expt}_0] \le \Pr[\mathsf{Expt}_1] + \frac{q_I^2 - q_I}{2(p-1)}. \tag{4}$$

$\mathsf{Expt}_1 \to \mathsf{Expt}_2$. In Expt_2 we replace the scalars $(\eta_i)_{i\in[n]}, (x_i)_{i\in[q_K]}, (r_{k,1})_{k\in[q_S]}$, a, b associated with group elements $(h_i)_{i\in[n]}, (\mathsf{pk}_i)_{i\in[q_K]}, (A'_k, B_k, \bar{A}_k)_{k\in[q_S]}, S$, $(T_i)_{i\in I_{\mathcal{P}}}$ with indeterminates $(\mathsf{H}_i)_{i\in[n]}, (\mathsf{X}_i)_{i\in[q_K]}, (\mathsf{R}_k)_{k\in[q_S]}, \mathsf{A}, \mathsf{B}$. Labels of group elements are now associated with rational functions in

$$\mathcal{R} \stackrel{\text{def}}{=} \mathbb{Z}_p(\mathsf{H}_1, \ldots, \mathsf{H}_n, \mathsf{X}_1, \ldots, \mathsf{X}_{q_K}, \mathsf{R}_1, \ldots, \mathsf{R}_{q_S}, \mathsf{A}, \mathsf{B})$$

instead of scalars in $\mathbb{Z}_p$. That is, we now define $\mathsf{Label}_i(f)$, where $f \in \mathcal{R}$, as follows (σ_i is initialized to $\emptyset$ as before; we also initialize a set F to $\{0\}$):

If $\sigma_i(f)$ is already defined, return $y := \sigma_i(f)$. Otherwise, choose a random, unused label $y \leftarrow \{0,1\}^{2\lceil \log p \rceil} \setminus \mathrm{range}(\sigma_i)$, set $\sigma_i(f) := y$, add f to F, and return y.

Note that set F contains all rational functions defined throughout the experiment. We assume each rational function f is stored in reduced canonical form, i.e., as the unique representation $f = \frac{P}{Q}$ where P, Q are coprime and Q is monic.

Expt_0 / $\boxed{\mathsf{Expt}_1}$:

1. $\sigma_1 := \emptyset$, $\sigma_2 := \emptyset$, $\sigma_T := \emptyset\boxed{,\ E := \emptyset}$.
2. $g_1 \leftarrow \mathsf{Label}_1(1)$, $g_2 \leftarrow \mathsf{Label}_2(1)$. For $i \in [n]$, $\eta_i \leftarrow \mathbb{Z}_p$ and $h_i \leftarrow \mathsf{Label}_1(\eta_i)$.
3. Run $\mathcal{A}(g_1, g_2, (h_i)_{i\in[n]})$ with access to oracles $\mathcal{O}\mathsf{Group}$, $\mathcal{O}\mathsf{Pair}$, $\mathcal{O}\mathsf{KeyGen}$, $\mathcal{O}\mathsf{Issue}$, $\mathcal{O}\mathsf{Issue}'$, $\mathcal{O}\mathsf{Show}$, $\mathcal{O}\mathsf{SetPolicy}$, $\mathcal{O}\mathsf{Vrfy}$.
4. $\mathcal{A}$ outputs $\mathcal{D}$, $(m_i)_{i\in\mathcal{D}}$, ctx, $A', \bar{B}, \bar{A}, \tilde{\sigma}$, and a ZKPoK of statement (2).
5. If $[A']_1 = 0$, verification of the ZKPoK with tag $(\mathcal{P}, \mathsf{pk}_{\mathcal{P}}, \mathsf{ctx})$ fails, or
$$[A']_1 \cdot \left(-1/a \cdot [\tilde{\sigma}]_2 + (\ell - 1)b + \textstyle\sum_{i\in I_{\mathcal{P}}} x_i\right) - [\bar{A}]_1 \neq 0,$$
return 0. (Recall $\mathcal{P}$ is the query to $\mathcal{O}\mathsf{SetPolicy}$.)
6. If $\mathcal{A}$ made a query $\mathcal{O}\mathsf{Issue}(i^*, \boldsymbol{m}^*)$ with $\mathsf{pk}_{i^*} \in \mathcal{P}$ and $m_i^* = m_i$ for all $i \in \mathcal{D}$, return 0.
7. If $\mathcal{A}$ made a query $\mathcal{O}\mathsf{Show}(j, \mathcal{P}, \mathsf{pk}_{\mathcal{P}}, \mathcal{D}, \mathsf{ctx})$ where $\mathsf{pk}_{i'_j} \in \mathcal{P}$ and $m'_{j,i} = m_i$ for all $i \in \mathcal{D}$, return 0.
8. If $(m_i)_{i\notin\mathcal{D}}, \tilde{e}, r_1, r_2$ cannot be extracted from the ZKPoK with
$$[\bar{A}]_1 + \tilde{e} \cdot [A']_1 - r_2 \cdot [\bar{B}]_1 = 0 \text{ and}$$
$$1 + \textstyle\sum_{i\in[n]} m_i\eta_i - 1/r_1 \cdot [\bar{B}]_1 = 0,$$
return 0. Otherwise, return 1.

$\mathcal{O}\mathsf{KeyGen}$: On the ith query, sample $x_i \leftarrow \mathbb{Z}_p$, and return $\mathsf{pk}_i \leftarrow \mathsf{Label}_2(x_i)$.

$\mathcal{O}\mathsf{Issue}(i_j, m_{j,1}, \ldots, m_{j,n})$: On the jth query, do:

1. $e_j \leftarrow \mathbb{Z}_p \setminus \{-x_{i_j}\}$. $\boxed{\text{If } e_j \in E\text{, abort. } E := E \cup \{e_j\}.}$
2. $A_j \leftarrow \mathsf{Label}_1\left(\frac{1+\sum_{i\in[n]} m_{j,i}\cdot\eta_i}{x_{i_j}+e_j}\right)$. Return (A_j, e_j).

$\mathcal{O}\mathsf{Issue}'(i'_j, \boldsymbol{m}'_j)$: On the jth query, $e'_j \leftarrow \mathbb{Z}_p \setminus \{-x_{i'_j}\}$, $\hat{A}_j \leftarrow \mathsf{Label}_1\left(\frac{1+\sum_{i\in[n]} m'_{j,i}\cdot\eta_i}{x_{i'_j}+e'_j}\right)$.

$\mathcal{O}\mathsf{Show}(j_k, \mathcal{P}_k, \mathsf{pk}_{\mathcal{P}_k}, \mathcal{D}_k, \mathsf{ctx}_k)$: On the kth query, do:

1. If $\mathsf{pk}_{i'_{j_k}} \notin \mathcal{P}_k$ or $\mathsf{AuditPolicy}(\mathcal{P}_k, \mathsf{pk}_{\mathcal{P}_k}) = 0$, return $\perp$.
2. Let $I := \{i \mid \mathsf{pk}'_i \in \mathcal{P}_k\}$. Parse $\mathsf{pk}_{\mathcal{P}_k}$ as $(S_k, \{T_{k,i}\}_{i\in I})$. $r_{k,1}, r_{k,2} \leftarrow \mathbb{Z}_p^*$, $t_k \leftarrow \mathbb{Z}_p$. $A'_k \leftarrow \mathsf{Label}_1\left(r_{k,1}r_{k,2} \cdot [\hat{A}_{j_k}]_1\right)$, $\bar{B}_k \leftarrow \mathsf{Label}_1\left(r_{k,1} \cdot (1 + \sum_{i\in[n]} m'_{j_k,i} \cdot \eta_i)\right)$, $\bar{A}_k \leftarrow \mathsf{Label}_1\left((-e'_{j_k} - t_k) \cdot [A'_k]_1 + r_{k,2} \cdot [\bar{B}_k]_1\right)$, and $\tilde{\sigma}_k \leftarrow \mathsf{Label}_2\left(t_k \cdot [S_k]_2 + \sum_{i\in I\setminus\{i'_{j_k}\}} [T_{k,i}]_2\right)$.
3. Return $A'_k, \bar{B}_k, \bar{A}_k, \tilde{\sigma}_k$, and a ZKPoK with tag $(\mathcal{P}_k, \mathsf{pk}_{\mathcal{P}_k}, \mathsf{ctx}_k)$ of statement (2).

$\mathcal{O}\mathsf{SetPolicy}(\mathcal{P} = \{\mathsf{pk}_i\}_{i\in I_{\mathcal{P}}})$:

1. $a \leftarrow \mathbb{Z}_p^*$, $b \leftarrow \mathbb{Z}_p$, and $S \leftarrow \mathsf{Label}_2(a)$. For $i \in I_{\mathcal{P}}$, $T_i \leftarrow \mathsf{Label}_2(x_i a + ba)$.
2. Return $\mathsf{pk}_{\mathcal{P}}$ as $(S, \{T_i\}_{i\in I_{\mathcal{P}}})$ and a ZKPoK of statement (1).

$\mathcal{O}\mathsf{Vrfy}(\mathcal{D}', (m'_i)_{i\in\mathcal{D}}, \mathsf{ctx}', \pi')$:

1. Parse π' as $A', \bar{B}, \bar{A}, \tilde{\sigma}$, and a ZKPoK of statement (2).
2. $f := [A']_1 \cdot \left(-1/a \cdot [\tilde{\sigma}]_2 + (\ell - 1)b + \sum_{i\in I_{\mathcal{P}}} x_i\right) - [\bar{A}]_1$.
3. If $[A']_1 = 0$, verification of the ZKPoK with tag $(\mathcal{P}, \mathsf{pk}_{\mathcal{P}}, \mathsf{ctx}')$ fails, or $f \neq 0$, return 0; else, return 1.

Fig. 1. Experiments Expt_0 and Expt_1. Changes from Expt_0 to Expt_1 are boxed.

Expt_2:

1. $\sigma_1 := \emptyset$, $\sigma_2 := \emptyset$, $\sigma_T := \emptyset$, $E := \emptyset$, $F := \{0\}$.
2. $g_1 \leftarrow \mathsf{Label}_1(1)$, $g_2 \leftarrow \mathsf{Label}_2(1)$. For $i \in [n]$, $h_i \leftarrow \mathsf{Label}_1(\mathsf{H}_i)$.
3. Run $\mathcal{A}(g_1, g_2, (h_i)_{i\in[n]})$ with access to oracles $\mathcal{O}\mathsf{Group}$, $\mathcal{O}\mathsf{Pair}$, $\mathcal{O}\mathsf{KeyGen}$, $\mathcal{O}\mathsf{Issue}$, $\mathcal{O}\mathsf{Issue}'$, $\mathcal{O}\mathsf{Show}$, $\mathcal{O}\mathsf{SetPolicy}$, $\mathcal{O}\mathsf{Vrfy}$.
4. $\mathcal{A}$ outputs $\mathcal{D}$, $(m_i)_{i\in\mathcal{D}}$, ctx, $A', \bar{B}, \bar{A}, \tilde{\sigma}$, and a ZKPoK of statement (2).
5. If $[A']_1 \equiv 0$, verification of the ZKPoK with tag $(\mathcal{P}, \mathsf{pk}_{\mathcal{P}}, \mathsf{ctx})$ fails, or
$$f_1 := [A']_1 \cdot \left(-\mathsf{A}^{-1} \cdot [\tilde{\sigma}]_2 + (\ell - 1)\mathsf{B} + \textstyle\sum_{i\in I_{\mathcal{P}}} \mathsf{X}_i\right) - [\bar{A}]_1 \not\equiv 0,$$
return 0. (Recall $\mathcal{P}$ is the query to $\mathcal{O}\mathsf{SetPolicy}$.)
6. If $\mathcal{A}$ made a query $\mathcal{O}\mathsf{Issue}(i^*, \boldsymbol{m}^*)$ with $\mathsf{pk}_{i^*} \in \mathcal{P}$ and $m_i^* = m_i$ for all $i \in \mathcal{D}$, return 0.
7. If $\mathcal{A}$ made a query $\mathcal{O}\mathsf{Show}(j, \mathcal{P}, \mathsf{pk}_{\mathcal{P}}, \mathcal{D}, \mathsf{ctx})$ where $\mathsf{pk}_{i'_j} \in \mathcal{P}$ and $m'_{j,i} = m_i$ for all $i \in \mathcal{D}$, return 0.
8. If $(m_i)_{i\notin\mathcal{D}}, \tilde{e}, r_1, r_2$ cannot be extracted from the ZKPoK with
$$f_2 := [\bar{A}]_1 + \tilde{e} \cdot [A']_1 - r_2 \cdot [\bar{B}]_1 \equiv 0 \text{ and}$$
$$f_3 := 1 + \textstyle\sum_{i\in[n]} m_i \mathsf{H}_i - 1/r_1 \cdot [\bar{B}]_1 \equiv 0,$$
return 0.
9. Set $F := F \cup \{f_1, f_2, f_3\}$. For $i \in [n]$, $\eta_i \leftarrow \mathbb{Z}_p$. For $i \in [q_K]$, $x_i \leftarrow \mathbb{Z}_p \setminus \left(\{-e_j\}_{j\in[q_I]\colon i_j=i} \cup \{-e'_{j_k}\}_{k\in[q_S]\colon i'_{j_k}=i}\right)$. For $k \in [q_S]$, $r_{k,1} \leftarrow \mathbb{Z}_p^*$. $a \leftarrow \mathbb{Z}_p^*$, $b \leftarrow \mathbb{Z}_p$. If any two rational functions in F evaluate to the same value at $(\eta_1, \ldots, \eta_n, x_1, \ldots, x_{q_K}, r_{1,1}, \ldots, r_{q_S,1}, a, b)$, return 0. Otherwise, return 1.

$\mathcal{O}\mathsf{KeyGen}$: On the ith query, return $\mathsf{pk}_i \leftarrow \mathsf{Label}_2(\mathsf{X}_i)$.

$\mathcal{O}\mathsf{Issue}(i_j, m_{j,1}, \ldots, m_{j,n})$: On the jth query, do:

1. $e_j \leftarrow \mathbb{Z}_p$. If $e_j \in E$, abort. $E := E \cup \{e_j\}$.
2. $A_j \leftarrow \mathsf{Label}_1\left(\frac{1+\sum_{i\in[n]} m_{j,i}\cdot \mathsf{H}_i}{\mathsf{X}_{i_j}+e_j}\right)$. Return (A_j, e_j).

$\mathcal{O}\mathsf{Issue}'(i'_j, \boldsymbol{m}'_j)$: On the jth query, $e'_j \leftarrow \mathbb{Z}_p$, $\hat{A}_j \leftarrow \mathsf{Label}_1\left(\frac{1+\sum_{i\in[n]} m'_{j,i}\cdot \mathsf{H}_i}{\mathsf{X}_{i'_j}+e'_j}\right)$.

$\mathcal{O}\mathsf{Show}(j_k, \mathcal{P}_k, \mathsf{pk}_{\mathcal{P}_k}, \mathcal{D}_k, \mathsf{ctx}_k)$: On the kth query, do:

1. If $\mathsf{pk}_{i'_{j_k}} \notin \mathcal{P}_k$ or $\mathsf{AuditPolicy}(\mathcal{P}_k, \mathsf{pk}_{\mathcal{P}_k}) = 0$, return $\perp$.
2. Let $I := \{i \mid \mathsf{pk}'_i \in \mathcal{P}_k\}$. Parse $\mathsf{pk}_{\mathcal{P}_k}$ as $(S_k, \{T_{k,i}\}_{i\in I})$. $r_{k,2} \leftarrow \mathbb{Z}_p^*$, $t_k \leftarrow \mathbb{Z}_p$. $A'_k \leftarrow \mathsf{Label}_1\left(\mathsf{R}_k r_{k,2} \cdot [\hat{A}_{j_k}]_1\right)$, $\bar{B}_k \leftarrow \mathsf{Label}_1\left(\mathsf{R}_k \cdot (1 + \sum_{i\in[n]} m'_{j_k,i} \cdot \mathsf{H}_i)\right)$, $\bar{A}_k \leftarrow \mathsf{Label}_1\left((-e'_{j_k} - t_k) \cdot [A'_k]_1 + r_{k,2} \cdot [\bar{B}_k]_1\right)$, and $\tilde{\sigma}_k \leftarrow \mathsf{Label}_2\left(t_k \cdot [S_k]_2 + \sum_{i\in I\setminus\{i'_{j_k}\}} [T_{k,i}]_2\right)$.
3. Return $A'_k, \bar{B}_k, \bar{A}_k, \tilde{\sigma}_k$, and a simulated ZKPoK with tag $(\mathcal{P}_k, \mathsf{pk}_{\mathcal{P}_k}, \mathsf{ctx}_k)$ of (2).

$\mathcal{O}\mathsf{SetPolicy}(\mathcal{P} = \{\mathsf{pk}_i\}_{i\in I_{\mathcal{P}}})$:

1. $S \leftarrow \mathsf{Label}_2(\mathsf{A})$. For $i \in I_{\mathcal{P}}$, $T_i \leftarrow \mathsf{Label}_2(\mathsf{X}_i\mathsf{A} + \mathsf{BA})$.
2. Return $\mathsf{pk}_{\mathcal{P}}$ as $(S, \{T_i\}_{i\in I_{\mathcal{P}}})$ and a simulated ZKPoK of statement (1).

$\mathcal{O}\mathsf{Vrfy}(\mathcal{D}', (m'_i)_{i\in\mathcal{D}}, \mathsf{ctx}', \pi')$:

1. Parse π' as $A', \bar{B}, \bar{A}, \tilde{\sigma}$, and a ZKPoK of statement (2).
2. $f := [A']_1 \cdot \left(-\mathsf{A}^{-1} \cdot [\tilde{\sigma}]_2 + (\ell - 1)\mathsf{B} + \sum_{i\in I_{\mathcal{P}}} \mathsf{X}_i\right) - [\bar{A}]_1$, $F := F \cup \{f\}$.
3. If $[A']_1 \equiv 0$, verification of the ZKPoK with tag $(\mathcal{P}, \mathsf{pk}_{\mathcal{P}}, \mathsf{ctx}')$ fails, or $f \not\equiv 0$, return 0; else, return 1.

Fig. 2. Experiment Expt_2 with changes from Expt_1 in blue.

Expt_2 defers sampling the values $(\eta_i)_{i\in[n]}, (x_i)_{i\in[q_K]}, (r_{k,1})_{k\in[q_S]}, a, b$ to the end, and returns 0 in line 9 if any two rational functions in F agree for the chosen values; otherwise, the view of the adversary in Expt_2 is equivalent to its view in Expt_1. As a consequence, we also make the following changes:

1. Verification of proofs now has to use rational function identities, affecting both $\mathcal{O}\mathsf{Vrfy}$ and the checks in lines 5 and 8. By initializing the set F to $\{0\}$, we account for these modified checks in the return condition of line 9.
2. The ZKPoK of statement (1) returned by $\mathcal{O}\mathsf{SetPolicy}$ and the ZKPoKs of statement (2) returned by $\mathcal{O}\mathsf{Show}$ now have to be simulated. Since we are assuming perfect zero knowledge, this is equivalent to computing the proofs honestly and thus leaves the adversary's view unchanged.
3. The values $\{e_j\}$ sampled in line 1 of $\mathcal{O}\mathsf{Issue}$ (and in $\mathcal{O}\mathsf{Issue}'$) can no longer be sampled as in Expt_1, where we first sample $x_i \leftarrow \mathbb{Z}_p$ and then (assuming no abort in line 1) distinct $e_j \leftarrow \mathbb{Z}_p \setminus \{-x_{i_j}\}$. In Expt_2, we reverse this order: we first sample distinct $e_j \leftarrow \mathbb{Z}_p$, and then $x_i \leftarrow \mathbb{Z}_p \setminus \{-e_j\}_{j\in[q_I]\colon i_j=i}$. It is not hard to see that these two distributions are equivalent.

It remains to bound the probability that any two rational functions in F coincide at $(\eta_1, \ldots, \eta_n, x_1, \ldots, x_{q_K}, r_{1,1}, \ldots, r_{q_S,1}, a, b)$. Toward this end, we count the total number of rational functions added to F throughout the experiment, bound the degrees of their numerators and denominators, and then apply Lemma 8.

The adversary is given a total of $q_I + 3q_S + n + 1$ elements in $\mathbb{G}_1$ (including g_1), and $q_K + q_S + \ell + 2$ elements in $\mathbb{G}_2$ (including g_2). Moreover, by making group-operation queries, it can produce at most $q_{\mathbb{G}}$ additional elements in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$. Finally, the pairing checks made in answering queries to $\mathcal{O}\mathsf{Vrfy}$ and the four success conditions in lines 5 and 8 correspond to at most $q_V + 4$ additional rational functions (the checks in $\mathcal{O}\mathsf{Vrfy}$ against the identity element of $\mathbb{G}_1$ and the same check in line 5 only need to be counted once, since the zero polynomial is in F). In total, we have at most

$$q_{\mathbb{G}} + q_K + q_I + 4q_S + q_V + \ell + n + 7$$

rational functions in F.

Next we consider the highest possible total degree of the numerators and denominators of those rational functions:

- Rational functions corresponding to elements in $\mathbb{G}_2$ are polynomials of total degree at most 2.
- Rational functions corresponding to elements in $\mathbb{G}_1$ have denominators of total degree at most $q_I + q_S$, the maximum degree of the least common multiple L of the $q_I + q_S$ linear factors in the denominators of the $(A_j)_{j\in[q_I]}$ and $(\hat{A}_{j_k})_{k\in[q_S]}$ from $\mathcal{O}\mathsf{Issue}$ and $\mathcal{O}\mathsf{Show}$ queries. Numerators of those rational functions have total degree at most $q_I + q_S + 2$ since L is multiplied by polynomials of degree at most 2 (due to the $\mathsf{R}_k\mathsf{H}_i$ terms from $\mathcal{O}\mathsf{Show}$ queries).
- For $\mathbb{G}_T$, we first consider group elements produced by the adversary using its group oracles. The numerator (resp., denominator) of the rational function

corresponding to any such element has total degree at most the sum of the maximum total degrees of the numerators (resp., denominators) of one rational function corresponding to an element in $\mathbb{G}_1$ and another in $\mathbb{G}_2$; this gives bounds of $q_I + q_S + 4$ for the numerator and $q_I + q_S$ for the denominator. There are also rational functions in F that correspond to elements in $\mathbb{G}_T$ not produced by the adversary, but that instead arise due to the checks in line 5 of the experiment and line 2 of $\mathcal{O}\mathsf{Vrfy}$. Those checks increase the denominator by one (due to the presence of $\mathsf{A}^{-1} \cdot [\tilde{\sigma}]_2$), giving a final bound of $q_I + q_S + 1$ for the degree of the denominator of rational functions corresponding to elements of $\mathbb{G}_T$.

In summary, we get bounds of $q_I + q_S + 4$ and $q_I + q_S + 1$ on the total degree of the numerator and denominator of rational functions in F, respectively.

Applying Lemma 8, the probability that any two such rational functions evaluate to the same value at the random point sampled in line 9 is at most

$$\frac{(2q_I + 2q_S + 5) \cdot p^{n+q_K+q_S+1}}{|\mathcal{S}|},$$

where $\mathcal{S} = \{(\eta_1, \ldots, \eta_n, x_1, \ldots, x_{q_K}, r_{1,1}, \ldots, r_{q_S,1}, a, b)\} \subset \mathbb{Z}_p^{n+q_K+q_S+2}$ with each η_i and b in $\mathbb{Z}_p$, each x_i in $\mathbb{Z}_p \setminus \left(\{-e_j\}_{j\in[q_I]\colon i_j=i} \cup \{-e'_{j_k}\}_{k\in[q_S]\colon i'_{j_k}=i}\right)$, and each $r_{i,1}$ and a in $\mathbb{Z}_p \setminus \{0\}$. Using

$$\begin{aligned}
|\mathcal{S}| &= p^{n+1} \cdot (p-1)^{q_S+1} \cdot \prod_{i\in[q_K]} \left(p - \left|\{-e_j\}_{j\in[q_I]\colon i_j=i} \cup \{-e'_{j_k}\}_{k\in[q_S]\colon i'_{j_k}=i}\right|\right) \\
&\geq p^{n+1} \cdot p^{q_S}(p - q_S - 1) \cdot p^{q_K-1}(p - q_I - q_S) \\
&\geq (p - q_I - 2q_S - 1) \cdot p^{n+q_K+q_S+1} \geq 1,
\end{aligned}$$

and taking a union bound over all pairs of rational functions in F, we get

$$\begin{aligned}
&\Pr[\mathsf{Expt}_1] \\
&\leq \Pr[\mathsf{Expt}_2] + \frac{(2q_I + 2q_S + 5) \cdot (q_{\mathbb{G}} + q_K + q_I + 4q_S + q_V + \ell + n + 7)^2}{2 \cdot (p - q_I - 2q_S - 1)}. \quad (5)
\end{aligned}$$

Expt_2. Having replaced all "secret" exponents by indeterminates, and assuming the $(e_j)_{j\in[q_I]}$ are distinct, we now show that $\Pr[\mathsf{Expt}_2] = 0$. Specifically, if $\mathsf{Vrfy}_{\mathsf{sk}_{\mathcal{P}}}(\mathcal{P}, \mathcal{D}, (m_i)_{i\in\mathcal{D}}, \mathsf{ctx}, \pi) = 1$ and the extraction in line 8 succeeds, then there exists $j^* \in [q_I]$ such that $\mathsf{pk}_{i_{j^*}} \in \mathcal{P}$ and $m_i = m_{j^*,i}$ for all $i \in \mathcal{D}$, and so $\mathcal{A}$'s forgery was trivial (and the check in line 6 would have failed).

The group elements given to the adversary (not including the outputs of $\mathcal{O}\mathsf{Group}$ and $\mathcal{O}\mathsf{Pair}$) correspond to the following rational functions:

- Inputs: $[g_1]_1 = 1$, $[g_2]_2 = 1$, $\left([h_i]_1 = \mathsf{H}_i\right)_{i\in[n]}$
- $\mathcal{O}\mathsf{KeyGen}$: $\left([\mathsf{pk}_i]_2 = \mathsf{X}_i\right)_{i\in[q_K]}$
- $\mathcal{O}\mathsf{Issue}$: $\left([A_j]_1 = \frac{1+\sum_{i\in[n]} m_{j,i}\mathsf{H}_i}{\mathsf{X}_{i_j}+e_j}\right)_{j\in[q_I]}$

– $\mathcal{O}\mathsf{Show}$: $\begin{pmatrix} [A'_k]_1 = \mathsf{R}_k r_{k,2} \cdot \frac{1+\sum_{i\in[n]} m'_{j_k,i}\mathsf{H}_i}{\mathsf{X}_{i'_{j_k}}+e'_{j_k}}, \\ [\bar{B}_k]_1 = \mathsf{R}_k \cdot (1+\sum_{i\in[n]} m'_{j_k,i}\mathsf{H}_i), \\ [\bar{A}_k]_1 = (-e'_{j_k}-t_k)\cdot[A'_k]_1 + r_{k,2}\cdot[\bar{B}_k]_1, \\ [\tilde{\sigma}_k]_2 = t_k\cdot[S_k]_2 + \sum_{i\in I\setminus\{i'_{j_k}\}}[T_{k,i}]_2 \end{pmatrix}_{k\in[q_S]}$

– $\mathcal{O}\mathsf{SetPolicy}$: $[S]_2 = \mathsf{A}$, $\left([T_i]_2 = \mathsf{X}_i\mathsf{A} + \mathsf{BA}\right)_{i\in I_{\mathcal{P}}}$.

Since $\mathcal{O}\mathsf{Group}$ only produces linear combinations of the rational functions above (that correspond to elements in either $\mathbb{G}_1$ or $\mathbb{G}_2$), and since $\mathcal{O}\mathsf{Pair}$ returns $\mathbb{G}_T$ elements, we can express the group elements $A', \bar{B}, \bar{A} \in \mathbb{G}_1, \tilde{\sigma} \in \mathbb{G}_2$ output by $\mathcal{A}$ as the following rational functions for some coefficients $\alpha, (\beta_i)_{i\in[n]}$, etc., where $\mathbf{H} = (\mathsf{H}_1, \ldots, \mathsf{H}_n)$ and $\mathbf{X} = (\mathsf{X}_1, \ldots, \mathsf{X}_{q_K})$:

$$[A']_1 = \alpha + \sum_{i\in[n]} \beta_i\mathsf{H}_i + \sum_{j\in[q_I]} \gamma_j \frac{1+\sum_{i\in[n]} m_{j,i}\mathsf{H}_i}{\mathsf{X}_{i_j}+e_j} + \sum_{k\in[q_S]} \mathsf{R}_k f_k(\mathbf{H},\mathbf{X}),$$
$$\text{where } f_k(\mathbf{H},\mathbf{X}) = \delta_k\left(1+\textstyle\sum_{i\in[n]} m'_{j_k,i}\mathsf{H}_i\right) + \varepsilon_k \frac{1+\sum_{i\in[n]} m'_{j_k,i}\mathsf{H}_i}{\mathsf{X}_{i'_{j_k}}+e'_{j_k}},$$

$$[\bar{A}]_1 = \alpha' + \sum_{i\in[n]} \beta'_i\mathsf{H}_i + \sum_{j\in[q_I]} \gamma'_j \frac{1+\sum_{i\in[n]} m_{j,i}\mathsf{H}_i}{\mathsf{X}_{i_j}+e_j} + \sum_{k\in[q_S]} \mathsf{R}_k f'_k(\mathbf{H},\mathbf{X}),$$
$$\text{where } f'_k(\mathbf{H},\mathbf{X}) = \delta'_k\left(1+\textstyle\sum_{i\in[n]} m'_{j_k,i}\mathsf{H}_i\right) + \varepsilon'_k \frac{1+\sum_{i\in[n]} m'_{j_k,i}\mathsf{H}_i}{\mathsf{X}_{i'_{j_k}}+e'_{j_k}},$$

$$[\bar{B}]_1 = \alpha'' + \sum_{i\in[n]} \beta''_i\mathsf{H}_i + \sum_{j\in[q_I]} \gamma''_j \frac{1+\sum_{i\in[n]} m_{j,i}\mathsf{H}_i}{\mathsf{X}_{i_j}+e_j} + \sum_{k\in[q_S]} \mathsf{R}_k f''_k(\mathbf{H},\mathbf{X}),$$
$$\text{where } f''_k(\mathbf{H},\mathbf{X}) = \delta''_k\left(1+\textstyle\sum_{i\in[n]} m'_{j_k,i}\mathsf{H}_i\right) + \varepsilon''_k \frac{1+\sum_{i\in[n]} m'_{j_k,i}\mathsf{H}_i}{\mathsf{X}_{i'_{j_k}}+e'_{j_k}},$$

$$[\tilde{\sigma}]_2 = \alpha''' + t\mathsf{A} + \sum_{i\in[q_K]} \gamma'''_i\mathsf{X}_i + \sum_{i\in I_{\mathcal{P}}} \delta'''_i(\mathsf{X}_i\mathsf{A} + \mathsf{BA}).$$

Moreover, verification passing (line 5) and extraction succeeding (line 8) correspond to the following identities of rational functions:

$$[A']_1 \not\equiv 0, \tag{6}$$
$$[\bar{A}]_1 \equiv [A']_1 \cdot \left(-\mathsf{A}^{-1}\cdot[\tilde{\sigma}]_2 + (\ell-1)\mathsf{B} + \textstyle\sum_{i\in I_{\mathcal{P}}} \mathsf{X}_i\right), \tag{7}$$
$$[\bar{A}]_1 \equiv -\tilde{e}\cdot[A']_1 + r_2\cdot[\bar{B}]_1, \tag{8}$$
$$1+\textstyle\sum_{i\in[n]} m_i\mathsf{H}_i \equiv 1/r_1\cdot[\bar{B}]_1. \tag{9}$$

Equation (9) corresponds to the rational function identity

$$r_1 + \sum_{i\in[n]} r_1 m_i \mathsf{H}_i \equiv [\bar{B}]_1$$

$$\equiv \alpha'' + \sum_{i\in[n]} \beta_i'' \mathsf{H}_i + \sum_{j\in[q_I]} \gamma_j'' \frac{1+\sum_{i\in[n]} m_{j,i}\mathsf{H}_i}{\mathsf{X}_{i_j}+e_j} + \sum_{k\in[q_S]} \mathsf{R}_k f_k''(\mathbf{H},\mathbf{X}).$$

Equating the constant terms, the terms involving (only) H_i, the terms involving (only) $\frac{1}{\mathsf{X}_{i_j}+e_j}$ (which are linearly independent since the $\{e_j\}$ are distinct; cf. Lemma 10 in Appendix C), and the terms involving R_k, this implies

$$\alpha'' = r_1, \quad \forall i : \beta_i'' = r_1 m_i, \quad \forall j : \gamma_j'' = 0, \quad \forall k : f_k''(\mathbf{H},\mathbf{X}) \equiv 0.$$

Equation (8) corresponds to

$$\begin{aligned}
&\alpha' + \sum_{i\in[n]} \beta_i' \mathsf{H}_i + \sum_{j\in[q_I]} \gamma_j' \frac{1+\sum_{i\in[n]} m_{j,i}\mathsf{H}_i}{\mathsf{X}_{i_j}+e_j} + \sum_{k\in[q_S]} \mathsf{R}_k f_k'(\mathbf{H},\mathbf{X}) \\
&\equiv -\tilde{e}\left(\alpha + \sum_{i\in[n]} \beta_i \mathsf{H}_i + \sum_{j\in[q_I]} \gamma_j \frac{1+\sum_{i\in[n]} m_{j,i}\mathsf{H}_i}{\mathsf{X}_{i_j}+e_j} + \sum_{k\in[q_S]} \mathsf{R}_k f_k(\mathbf{H},\mathbf{X})\right) \\
&\quad + r_2\left(r_1 + \sum_{i\in[n]} r_1 m_i \mathsf{H}_i\right) \\
&\equiv r_1 r_2 - \tilde{e}\alpha + \sum_{i\in[n]} (r_1 r_2 m_i - \tilde{e}\beta_i)\mathsf{H}_i + \sum_{j\in[q_I]} -\tilde{e}\gamma_j \frac{1+\sum_{i\in[n]} m_{j,i}\mathsf{H}_i}{\mathsf{X}_{i_j}+e_j} \\
&\quad + \sum_{k\in[q_S]} -\tilde{e}\mathsf{R}_k f_k(\mathbf{H},\mathbf{X}).
\end{aligned}$$

Equating terms as before, this implies

$$\alpha' = r_1 r_2 - \tilde{e}\alpha, \quad \forall i : \beta_i' = r_1 r_2 m_i - \tilde{e}\beta_i, \quad \forall j : \gamma_j' = -\tilde{e}\gamma_j, \tag{10}$$

$$\forall k : f_k'(\mathbf{H},\mathbf{X}) \equiv -\tilde{e} f_k(\mathbf{H},\mathbf{X}). \tag{11}$$

Equation (7) corresponds to

$$\begin{aligned}
&\alpha' + \sum_{i\in[n]} \beta_i' \mathsf{H}_i + \sum_{j\in[q_I]} \gamma_j' \frac{1+\sum_{i\in[n]} m_{j,i}\mathsf{H}_i}{\mathsf{X}_{i_j}+e_j} + \sum_{k\in[q_S]} \mathsf{R}_k f_k'(\mathbf{H},\mathbf{X}) \\
&\equiv \left(\alpha + \sum_{i\in[n]} \beta_i \mathsf{H}_i + \sum_{j\in[q_I]} \gamma_j \frac{1+\sum_{i\in[n]} m_{j,i}\mathsf{H}_i}{\mathsf{X}_{i_j}+e_j} + \sum_{k\in[q_S]} \mathsf{R}_k f_k(\mathbf{H},\mathbf{X})\right) \\
&\quad \cdot \left(-t - \sum_{i\in I_{\mathcal{P}}} \delta_i'''(\mathsf{X}_i + \mathsf{B}) - (\alpha''' + \sum_{i\in[q_K]} \gamma_i''' \mathsf{X}_i)\mathsf{A}^{-1} + (\ell-1)\mathsf{B} + \sum_{i\in I_{\mathcal{P}}} \mathsf{X}_i\right).
\end{aligned}$$

By Equation (6), the first factor on the right-hand side is nonzero. Thus, looking at the coefficients of A^{-1}, we see that $\alpha''' + \sum_i \gamma_i''' \mathsf{X}_i \equiv 0$, which in turn implies

$$\alpha''' = 0 \text{ and } \forall i : \gamma_i''' = 0.$$

Looking at the coefficients of B, we must have $\sum_{i\in I_{\mathcal{P}}} \delta_i''' = \ell - 1$, or equivalently

$$\sum_{i\in I_{\mathcal{P}}} (1 - \delta_i''') = 1. \tag{12}$$

In particular, this means there is at least one value of $i \in I_{\mathcal{P}}$, call it i^*, for which $1 - \delta_{i^*}''' \neq 0$. Simplifying, and then plugging in Equation (11), we obtain

$$\begin{aligned}
&\alpha' + \sum_{i\in[n]} \beta_i' \mathsf{H}_i + \sum_{j\in[q_I]} \gamma_j' \frac{1 + \sum_{i\in[n]} m_{j,i}\mathsf{H}_i}{\mathsf{X}_{i_j} + e_j} + \sum_{k\in[q_S]} -\tilde{e}\mathsf{R}_k f_k(\mathbf{H},\mathbf{X}) \\
&\equiv \left(\alpha + \sum_{i\in[n]} \beta_i \mathsf{H}_i + \sum_{j\in[q_I]} \gamma_j \frac{1 + \sum_{i\in[n]} m_{j,i}\mathsf{H}_i}{\mathsf{X}_{i_j} + e_j} + \sum_{k\in[q_S]} \mathsf{R}_k f_k(\mathbf{H},\mathbf{X})\right) \\
&\quad \cdot \left(-t + \sum_{i\in I_{\mathcal{P}}} (1 - \delta_i''')\mathsf{X}_i\right).
\end{aligned}$$

When multiplied out, the right-hand side has terms $\sum_{i\in I_{\mathcal{P}}} \beta_j(1 - \delta_i''') \cdot \mathsf{X}_i\mathsf{H}_j$, and there are no terms of the form $\mathsf{X}_i\mathsf{H}_j$ on the left-hand side. Since $1 - \delta_{i^*}''' \neq 0$, we therefore must have $\beta_j = 0$ for all $j \in [n]$. A similar argument applies to the terms linear in the $\{\mathsf{X}_i\}$, implying $\alpha = 0$. Summarizing,

$$\alpha = 0 \text{ and } \forall i : \beta_i = 0. \tag{13}$$

Moreover, fixing an arbitrary $k \in [q_S]$ and equating terms in R_k, we get

$$\begin{aligned}
-\tilde{e} f_k(\mathbf{H},\mathbf{X}) &\equiv f_k(\mathbf{H},\mathbf{X}) \left(-t + \sum_{i\in I_{\mathcal{P}}} (1 - \delta_i''')\mathsf{X}_i\right) \\
\iff \qquad 0 &\equiv f_k(\mathbf{H},\mathbf{X}) \left(\tilde{e} - t + \sum_{i\in I_{\mathcal{P}}} (1 - \delta_i''')\mathsf{X}_i\right).
\end{aligned}$$

Since $1 - \delta_{i^*}''' \neq 0$, the polynomial $\tilde{e} - t + \sum_{i\in I_{\mathcal{P}}} (1 - \delta_i''')\mathsf{X}_i$ is nonzero. Since k was arbitrary, we conclude that

$$\forall k : f_k(\mathbf{H},\mathbf{X}) \equiv 0.$$

Simplifying again, we have

$$\begin{aligned}
&\alpha' + \sum_{i\in[n]} \beta_i' \mathsf{H}_i + \sum_{j\in[q_I]} \gamma_j' \frac{1 + \sum_{i\in[n]} m_{j,i}\mathsf{H}_i}{\mathsf{X}_{i_j} + e_j} \\
&\equiv \left(\sum_{j\in[q_I]} \gamma_j \frac{1 + \sum_{i\in[n]} m_{j,i}\mathsf{H}_i}{\mathsf{X}_{i_j} + e_j}\right) \cdot \left(-t + \sum_{i\in I_{\mathcal{P}}} (1 - \delta_i''')\mathsf{X}_i\right).
\end{aligned}$$

By Equation (6), the first factor on the right-hand side satisfies $[A']_1 \not\equiv 0$. Thus, we must have $q_I > 0$ and at least one index $j \in [q_I]$, call it j^*, for which $\gamma_{j^*} \neq 0$.

We noted earlier that at least one term in the second summation on the right-hand side (namely, the term involving X_{i^*}) is nonzero. If the first summation on the right-hand side has any term involving X_{i_j}, $i_j \neq i^*$, then we end up with some term of the form $\frac{\mathsf{X}_{i^*}}{\mathsf{X}_{i_j}+e_j}$, $i_j \neq i^*$, which has no counterpart on the left-hand side. We may thus conclude that $\gamma_j = \gamma'_j = 0$ if $i_j \neq i^*$, and so, in particular, $i_{j^*} = i^*$. Arguing as before but now about cross terms of the form $\frac{\mathsf{X}_i}{\mathsf{X}_{i_{j^*}}+e_{j^*}}$, we see that $1-\delta'''_i = 0$ for all $i \neq i^*$, and hence $1-\delta'''_{i^*} = 1$ (cf. Equation (12)). Thus, the above simplifies to

$$\begin{aligned}
&\alpha' + \sum_{i\in[n]} \beta'_i \mathsf{H}_i + \sum_{j\in[q_I]\colon\, i_j=i^*} \gamma'_j \frac{1+\sum_{i\in[n]} m_{j,i}\mathsf{H}_i}{\mathsf{X}_{i^*}+e_j} \\
&\equiv \left(\sum_{j\in[q_I]\colon\, i_j=i^*} \gamma_j \frac{1+\sum_{i\in[n]} m_{j,i}\mathsf{H}_i}{\mathsf{X}_{i^*}+e_j}\right)\cdot(-t+\mathsf{X}_{i^*}) \\
&\equiv \sum_{j\in[q_I]\colon\, i_j=i^*} \gamma_j \cdot \left(1+\sum\nolimits_{i\in[n]} m_{j,i}\mathsf{H}_i\right)\cdot \frac{\mathsf{X}_{i^*}-t}{\mathsf{X}_{i^*}+e_j} \\
&\equiv \sum_{j\in[q_I]\colon\, i_j=i^*} \gamma_j \cdot \left(1+\sum\nolimits_{i\in[n]} m_{j,i}\mathsf{H}_i\right)\cdot \frac{\mathsf{X}_{i^*}+e_j-e_j-t}{\mathsf{X}_{i^*}+e_j} \\
&\equiv \sum_{j\in[q_I]\colon\, i_j=i^*} \gamma_j \cdot \left(1+\sum\nolimits_{i\in[n]} m_{j,i}\mathsf{H}_i\right)\cdot \left(1+\frac{-e_j-t}{\mathsf{X}_{i^*}+e_j}\right) \\
&\equiv \sum_{j\in[q_I]\colon\, i_j=i^*} \left(\gamma_j + \gamma_j \sum_{i\in[n]} m_{j,i}\mathsf{H}_i + \gamma_j(-e_j-t)\left(\frac{1+\sum_{i\in[n]} m_{j,i}\mathsf{H}_i}{\mathsf{X}_{i^*}+e_j}\right)\right) \\
&\equiv \sum_{\substack{j\in[q_I]\colon\\ i_j=i^*}} \left(\gamma_j + \sum_{i\in[n]} \gamma_j m_{j,i}\mathsf{H}_i\right) + \sum_{\substack{j\in[q_I]\colon\\ i_j=i^*}} \gamma_j(-e_j-t)\left(\frac{1+\sum_{i\in[n]} m_{j,i}\mathsf{H}_i}{\mathsf{X}_{i^*}+e_j}\right).
\end{aligned}$$

If there were two distinct indices j_1^*, j_2^* for which $\gamma_{j_1^*}, \gamma_{j_2^*} \neq 0$, then (equating terms in $\frac{1}{\mathsf{X}_{i^*}+e_j}$) we would get

$$\gamma'_{j_1^*} = \gamma_{j_1^*}(-e_{j_1^*}-t) \quad \text{and} \quad \gamma'_{j_2^*} = \gamma_{j_2^*}(-e_{j_2^*}-t).$$

Combined with the fact that $\gamma'_j = -\tilde{e}\gamma_j$ for all j (cf. Equation (10)), this would imply $e_{j_1^*} = e_{j_2^*}$, contradicting our assumption that the $\{e_j\}$ are distinct. We conclude that $\gamma_j = \gamma'_j = 0$ for all $j \neq j^*$, and obtain

$$\begin{aligned}
&\alpha' + \sum_{i\in[n]} \beta'_i \mathsf{H}_i + \gamma'_{j^*} \frac{1+\sum_{i\in[n]} m_{j^*,i}\mathsf{H}_i}{\mathsf{X}_{i_{j^*}}+e_{j^*}} \\
&\equiv \gamma_{j^*} + \sum_{i\in[n]} \gamma_{j^*} m_{j^*,i}\mathsf{H}_i + \gamma_{j^*}(-e_{j^*}-t)\left(\frac{1+\sum_{i\in[n]} m_{j^*,i}\mathsf{H}_i}{\mathsf{X}_{i^*}+e_{j^*}}\right).
\end{aligned}$$

Equating terms, we get $\alpha' = \gamma_{j^*} \neq 0$ and $\beta'_i = \gamma_{j^*} m_{j^*,i} = \alpha' m_{j^*,i}$. Plugging $\alpha = 0$ and $\beta_i = 0$ from Equation (13) into $\alpha' = r_1 r_2 - \tilde{e}\alpha$ and $\beta'_i = r_1 r_2 m_i - \tilde{e}\beta_i$ from Equation (10) gives $\beta'_i = \alpha' m_i$. Thus, $(m_1, \ldots, m_n) = (m_{j^*,1}, \ldots, m_{j^*,n})$, which together with $i_{j^*} = i^* \in I_{\mathcal{P}}$ means $\mathcal{A}$'s forgery was trivial.

Combining Equations (3) to (5) with $\Pr[\mathsf{Expt}_2] = 0$ yields the theorem. □

6 Conclusion and Open Questions

We show how to achieve issuer hiding for BBS-based anonymous credentials. Our work can easily be integrated into existing standards [19,30,25,18], and has efficiency advantages compared to prior work.

Several interesting directions remain open. For one, both our work and that of Sanders and Traoré rely on the GGM. Yet BBS anonymous credentials can be proven secure without that, so it would be nice to avoid it here as well. Another interesting question is whether it is possible to remove the requirement for policy keys at all (while still maintaining acceptable efficiency), or at least whether it is possible to have policy keys of constant size. Finally, it would be useful to add issuer hiding to post-quantum anonymous-credential schemes.

Acknowledgments

Work of the second author was funded by the Vienna Science and Technology Fund (WWTF) [10.47379/VRG18002] and the Austrian Science Fund (FWF) [10.55776/F8515-N]. We thank the anonymous reviewers of PKC 2026 for their helpful comments.

References

1. Au, M.H., Susilo, W., Mu, Y.: Constant-size dynamic k-TAA. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 111–125. Springer (2006). https://doi.org/10.1007/11832072_8
2. Bicer, O., Küpçü, A.: Versatile ABS: Usage limited, revocable, threshold traceable, authority hiding, decentralized attribute based signatures. Cryptology ePrint Archive, Report 2019/203 (2019), https://eprint.iacr.org/2019/203
3. Bobolz, J., Eidens, F., Krenn, S., Ramacher, S., Samelin, K.: Issuer-hiding attribute-based credentials. In: CANS 2021. LNCS, vol. 13099, pp. 158–178. Springer (Dec 2021). https://doi.org/10.1007/978-3-030-92548-2_9
4. Boneh, D., Boyen, X., Goh, E.J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) Eurocrypt 2005. LNCS, vol. 3494, pp. 440–456. Springer (May 2005). https://doi.org/10.1007/11426639_26
5. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) Crypto 2004. LNCS, vol. 3152, pp. 41–55. Springer (Aug 2004). https://doi.org/10.1007/978-3-540-28628-8_3
6. Bosk, D., Frey, D., Gestin, M., Piolle, G.: Hidden issuer anonymous credential. PoPETs **2022**(4), 571–607 (Oct 2022). https://doi.org/10.56553/popets-2022-0123

7. Camenisch, J., Drijvers, M., Lehmann, A.: Anonymous attestation using the strong Diffie Hellman assumption revisited. In: Franz, M., Papadimitratos, P. (eds.) TRUST 2016. LNCS, vol. 9824, pp. 1–20. Springer, Cham (Aug 2016). https://doi.org/10.1007/978-3-319-45572-3_1
8. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) Crypto 2004. LNCS, vol. 3152, pp. 56–72. Springer (Aug 2004). https://doi.org/10.1007/978-3-540-28628-8_4
9. Chase, M., Meiklejohn, S., Zaverucha, G.: Algebraic MACs and keyed-verification anonymous credentials. In: Ahn, G.J., Yung, M., Li, N. (eds.) ACM CCS 2014. pp. 1205–1216. ACM Press (Nov 2014). https://doi.org/10.1145/2660267.2660328
10. Connolly, A., Deschamps, J., Lafourcade, P., Perez-Kempner, O.: Protego: Efficient, revocable and auditable anonymous credentials with applications to hyperledger fabric. In: Isobe, T., Sarkar, S. (eds.) Indocrypt 2022. LNCS, vol. 13774, pp. 249–271. Springer (Dec 2022). https://doi.org/10.1007/978-3-031-22912-1_11
11. Connolly, A., Lafourcade, P., Perez-Kempner, O.: Improved constructions of anonymous credentials from structure-preserving signatures on equivalence classes. In: PKC 2022, Part I. LNCS, vol. 13177, pp. 409–438. Springer (2022). https://doi.org/10.1007/978-3-030-97121-2_15
12. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: Kilian, J. (ed.) Crypto 2001. LNCS, vol. 2139, pp. 566–598. Springer (Aug 2001). https://doi.org/10.1007/3-540-44647-8_33
13. Elkhiyaoui, K., De Caro, A., Androulaki, E.: Multi-issuer anonymous credentials without a root authority. Cryptology ePrint Archive, Report 2021/1669 (2021), https://eprint.iacr.org/2021/1669
14. European Commission: The European digital identity wallet architecture and reference framework (2023), https://digital-strategy.ec.europa.eu/en/library/european-digital-identity-wallet-architecture-and-reference-framework, see also https://github.com/eu-digital-identity-wallet/eudi-doc-architecture-and-reference-framework/issues/200
15. Faust, S., Kohlweiss, M., Marson, G.A., Venturi, D.: On the non-malleability of the Fiat-Shamir transform. In: Galbraith, S.D., Nandi, M. (eds.) Indocrypt 2012. LNCS, vol. 7668, pp. 60–79. Springer (Dec 2012). https://doi.org/10.1007/978-3-642-34931-7_5
16. Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) Asiacrypt 2006. LNCS, vol. 4284, pp. 444–459. Springer (Dec 2006). https://doi.org/10.1007/11935230_29
17. ISO/IEC: Information technology—security techniques—anonymous digital signatures, part 2: Mechanisms using a group public key (2013), https://www.iso.org/standard/56916.html, ISO/IEC 20008-2:2013
18. Kalos, V., Bernstein, G.: Blind BBS signatures (2024), https://datatracker.ietf.org/doc/draft-kalos-bbs-blind-signatures/03
19. Looker, T., Kalos, V., Whitehead, A., Lodder, M.: The BBS signature scheme (2025), https://www.ietf.org/archive/id/draft-irtf-cfrg-bbs-signatures-09.html, Internet Draft
20. Maurer, U.M.: Abstract models of computation in cryptography (invited paper). In: Smart, N.P. (ed.) 10th IMA Intl. Conference on Cryptography and Coding. LNCS, vol. 3796, pp. 1–12. Springer (2005). https://doi.org/10.1007/11586821_1
21. Mir, O., Bauer, B., Griffy, S., Lysyanskaya, A., Slamanig, D.: Aggregate signatures with versatile randomization and issuer-hiding multi-authority anonymous credentials. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) ACM CCS 2023. pp. 30–44. ACM Press (Nov 2023). https://doi.org/10.1145/3576915.3623203

22. Orrù, M.: Revisiting keyed-verification anonymous credentials. Cryptology ePrint Archive, Report 2024/1552 (2024), https://eprint.iacr.org/2024/1552
23. Pointcheval, D., Sanders, O.: Short randomizable signatures. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 111–126. Springer (Feb / Mar 2016). https://doi.org/10.1007/978-3-319-29485-8_7
24. Sanders, O., Traoré, J.: Compact issuer-hiding authentication, application to anonymous credential. PoPETs **2024**(3), 645–658 (Jul 2024). https://doi.org/10.56553/popets-2024-0097
25. Schlesinger, S., Katz, J.: Anonymous credit tokens (2025), https://www.ietf.org/archive/id/draft-schlesinger-cfrg-act-00.html, Internet Draft
26. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) Eurocrypt '97. LNCS, vol. 1233, pp. 256–266. Springer (May 1997). https://doi.org/10.1007/3-540-69053-0_18
27. Tessaro, S., Zhu, C.: Revisiting BBS signatures. In: Hazay, C., Stam, M. (eds.) Eurocrypt 2023, Part V. LNCS, vol. 14008, pp. 691–721. Springer (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_24
28. W3C: Verifiable credentials data model v2.0 (2025), https://www.w3.org/TR/vc-data-model-2.0
29. Yun, C., Wood, C.A., Raykova, M., Schlesinger, S.: Anonymous tokens with hidden metadata (2025), https://datatracker.ietf.org/doc/html/draft-yun-cfrg-athm-00
30. Yun, C., Wood, C.: Anonymous rate-limited credentials (2025), https://datatracker.ietf.org/doc/html/draft-yun-privacypass-crypto-arc-00

A ZKPoKs

We describe ZKPoKs for statements (1) and (2) constructed from standard discrete logarithm-based Σ-protocols using the Fiat–Shamir transform. We let $H\colon \{0,1\}^* \to \mathbb{Z}_p$ be a hash function that we model as a random oracle. Results of Faust et al. [15] show that these are simulation extractable.

For the first statement, we have public values $\{\mathsf{pk}_i\}_{i\in[\ell]}, S, \{T_i\}_{i\in[\ell]}$; the prover wants to prove

$$\exists a, b \text{ s.t. } S = g_2^a \ \wedge\ \forall i : T_i = (\mathsf{pk}_i \cdot g_2^b)^a,$$

and knows a witness a, b satisfying the above. Since $S \neq 1$ in our application, a is invertible and the above is equivalent to

$$\exists a', b \text{ s.t. } S^{a'} = g_2 \ \wedge\ \forall i : T_i^{a'} g_2^{-b} = \mathsf{pk}_i,$$

with $a' = a^{-1}$. The prover computes a proof of this statement as follows:

1. Choose $r_a, r_b \leftarrow \mathbb{Z}_p$ and compute $X_S := S^{r_a}$ and $X_i := T_i^{r_a} g_2^{-r_b}$ for all i.
2. Compute $\gamma := H\left((\mathsf{pk}_i)_{i\in[\ell]}, S, (T_i)_{i\in[\ell]}, X_S, (X_i)_{i\in[\ell]}\right)$.
3. Set $\bar{a} := \gamma a' + r_a$ and $\bar{b} := \gamma b + r_b$.
4. Output the proof $(\gamma, \bar{a}, \bar{b})$.

To verify a proof $(\gamma, \bar{a}, \bar{b})$, check that

$$H\left((\mathsf{pk}_i)_{i\in[\ell]}, S, (T_i)_{i\in[\ell]}, S^{\bar{a}} g_2^{-\gamma}, \left(T_i^{\bar{a}} g_2^{-\bar{b}} \mathsf{pk}_i^{-\gamma}\right)_{i\in[\ell]}\right) \stackrel{?}{=} \gamma.$$

For the second statement, we have public values $\boldsymbol{h} = (h_i)_{i\in[n]}$, $\mathcal{D}$, $(m_i)_{i\in\mathcal{D}}$, A', $\bar{B}$, $\bar{A}$; the prover wants to prove

$$\begin{aligned}&\exists (m_i)_{i\notin\mathcal{D}}, \tilde{e}, r_1, r_2 \text{ s.t.}\\ &(A')^{-\tilde{e}} \cdot \bar{B}^{r_2} = \bar{A} \ \wedge\ \bar{B}^{1/r_1} \cdot \prod_{i\notin\mathcal{D}} h_i^{-m_i} = g_1 \cdot \prod_{i\in\mathcal{D}} h_i^{m_i}\end{aligned}$$

using $(\mathcal{P}, \mathsf{pk}_{\mathcal{P}}, \mathsf{ctx})$ as a tag, and knows a witness $(m_i)_{i\notin\mathcal{D}}, \tilde{e}, r_1, r_2$ satisfying the above. The prover computes a proof of this statement as follows:

- Choose $s_e, s_1, s_2, (s_i')_{i\notin\mathcal{D}} \leftarrow \mathbb{Z}_p$; then compute $X_A \coloneqq (A')^{-s_e} \cdot \bar{B}^{s_2}$ and

$$X_B \coloneqq \bar{B}^{s_1} \cdot \prod_{i\notin\mathcal{D}} h_i^{-s_i'}.$$

- Compute $\gamma \coloneqq H\left(\mathcal{P}, \mathsf{pk}_{\mathcal{P}}, \mathsf{ctx}, \mathcal{D}, (m_i)_{i\in\mathcal{D}}, A', \bar{B}, \bar{A}, X_A, X_B\right)$.
- Set $\bar{e} \coloneqq \gamma\tilde{e} + s_e$, $\bar{r}_1 \coloneqq \gamma/r_1 + s_1$, $\bar{r}_2 \coloneqq \gamma r_2 + s_2$, and $(\bar{m}_i \coloneqq \gamma m_i + s_i')_{i\notin\mathcal{D}}$.
- Output the proof $(\gamma, \bar{e}, \bar{r}_1, \bar{r}_2, (\bar{m}_i)_{i\notin\mathcal{D}})$.

To verify a proof $(\gamma, \bar{e}, \bar{r}_1, \bar{r}_2, (\bar{m}_i)_{i\notin\mathcal{D}})$, set $B' \coloneqq g_1 \cdot \prod_{i\in\mathcal{D}} h_i^{m_i}$ and check that

$$H\left(\mathcal{P}, \mathsf{pk}_{\mathcal{P}}, \mathsf{ctx}, \mathcal{D}, (m_i)_{i\in\mathcal{D}}, A', \bar{B}, \bar{A}, (A')^{-\bar{e}}\bar{B}^{\bar{r}_2}\bar{A}^{-\gamma}, \bar{B}^{\bar{r}_1}\prod_{i\notin\mathcal{D}} h_i^{-\bar{m}_i}(B')^{-\gamma}\right) \stackrel{?}{=} \gamma.$$

B A Potential Attack on Issuer-Hiding Anonymity

We show an attack on issuer-hiding anonymity if the policy key is malformed and the ZKPoK of well-formedness is omitted (or not verified). For simplicity, we fix $n = 1$ and write h in place of h_1. Let $\mathsf{pk}_0 = g_2^{x_0}$ and $\mathsf{pk}_1 = g_2^{x_1}$ be two public keys with $x_0 \neq x_1$. Fix an attribute $m = m_1$, let $B = g_1 h^m$, and let (A_0, e_0) and (A_1, e_1) be honestly generated credentials with respect to each of the public keys, so $A_0 = B^{1/(x_0+e_0)}$ and $A_1 = B^{1/(x_1+e_1)}$.

Next consider a malicious verifier who sets $\mathcal{P} \coloneqq \{\mathsf{pk}_0, \mathsf{pk}_1\}$, $S \coloneqq g_2^a$, $T_0 \coloneqq (\mathsf{pk}_0 \cdot g_2^{b_0})^a$, and $T_1 \coloneqq (\mathsf{pk}_1 \cdot g_2^{b_1})^a$ for arbitrary $a \neq 0$ and $b_0 \neq b_1$. We show that if a client uses this policy key then the verifier can distinguish a proof generated using (A_0, e_0) from one generated using (A_1, e_1). Concretely, given a proof containing $A', \bar{B}, \bar{A}$, and $\tilde{\sigma}$, the verifier knows that (A_0, e_0) was used iff

$$\mathsf{e}(A', (\tilde{\sigma} \cdot T_1^{-1})^{-1/a}) \cdot \mathsf{e}(A', \mathsf{pk}_0) = \mathsf{e}(\bar{A}, g_2).$$

To see this, note that if (A_0, e_0) was used to generate the proof then

$$\begin{aligned}\mathsf{e}(A', (\tilde{\sigma} \cdot T_1^{-1})^{-1/a}) \cdot \mathsf{e}(A', \mathsf{pk}_0) &= \mathsf{e}(A', (S^t T_1 T_1^{-1})^{-1/a}) \cdot \mathsf{e}((A')^{-e_0}\bar{B}^{r_2}, g_2)\\ &= \mathsf{e}(A', g_2^{-t}) \cdot \mathsf{e}((A')^{-e_0}\bar{B}^{r_2}, g_2)\\ &= \mathsf{e}((A')^{-t}(A')^{-e_0}\bar{B}^{r_2}, g_2)\\ &= \mathsf{e}(\bar{A}, g_2).\end{aligned}$$

On the other hand, if (A_1, e_1) was used to generate the proof then

$$\begin{aligned}
\mathsf{e}(A', (\tilde{\sigma} \cdot T_1^{-1})^{-1/a}) \cdot \mathsf{e}(A', \mathsf{pk}_0) &= \mathsf{e}(A', (S^t T_0 T_1^{-1})^{-1/a}) \cdot \mathsf{e}(A', g_2^{x_0}) \\
&= \mathsf{e}(A', g_2^{-t-x_0-b_0+x_1+b_1}) \cdot \mathsf{e}(A', g_2^{x_0}) \\
&= \mathsf{e}((A')^{-t+x_1+b_1-b_0}, g_2) \\
&\neq \mathsf{e}((A')^{-t+x_1}, g_2) \\
&= \mathsf{e}((A')^{-t-e_1}(A')^{x_1+e_1}, g_2) \\
&= \mathsf{e}((A')^{-e_1-t}\bar{B}^{r_2}, g_2) = \mathsf{e}(\bar{A}, g_2).
\end{aligned}$$

C Linear Independence of Rational Functions

We prove the following lemma.

Lemma 10. *Let p be prime. The following set of rational functions is linearly independent:*

$$\left\{ \frac{1}{\mathsf{X} - i} \;\middle|\; i \in \mathbb{Z}_p \right\} \subset \mathbb{Z}_p(\mathsf{X}).$$

Proof. Fix an arbitrary vector $\boldsymbol{\alpha} = (\alpha_i)_{i \in \mathbb{Z}_p} \in \mathbb{Z}_p^p$. We will show that if

$$\sum_{i \in \mathbb{Z}_p} \frac{\alpha_i}{\mathsf{X} - i} \equiv 0,$$

then $\boldsymbol{\alpha} = \mathbf{0}$. The above is equivalent to

$$\frac{\sum_{i \in \mathbb{Z}_p} \alpha_i \prod_{j \in \mathbb{Z}_p \setminus \{i\}} (\mathsf{X} - j)}{\prod_{j \in \mathbb{Z}_p} (\mathsf{X} - j)} \equiv 0,$$

which implies

$$\sum_{i \in \mathbb{Z}_p} \alpha_i \prod_{j \in \mathbb{Z}_p \setminus \{i\}} (\mathsf{X} - j) \equiv 0.$$

Plugging in any $i^* \in \mathbb{Z}_p$ for X, we get

$$\sum_{i \in \mathbb{Z}_p} \alpha_i \prod_{j \in \mathbb{Z}_p \setminus \{i\}} (i^* - j) = \alpha_{i^*} \prod_{j \in \mathbb{Z}_p \setminus \{i^*\}} (i^* - j) = 0.$$

Since $\prod_{j \in \mathbb{Z}_p \setminus \{i^*\}} (i^* - j) \neq 0$, this implies that $\alpha_{i^*} = 0$, finishing the proof. □